SILESIAN UNIVERSITY OF TECHNOLOGY

FACULTY OF MECHANICAL ENGINEERING

DEPARTMENT OF THEORETICAL AND APPLIED MECHANICS

A COMPUTATIONAL INTELLIGENCE FRAMEWORK FOR PDE-CONSTRAINED TOPOLOGY OPTIMIZATION

Doctoral Dissertation

M.Sc. Eng., Nikhil TATKE

Supervisor:

PhD, DSc, Eng. Jarosław KACZMARCZYK, Prof. SUT

In Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Gliwice, 2026

Contents

Abstract

Structural topology optimization seeks the optimal distribution of material within a design domain under equilibrium constraints and prescribed performance objectives. In density-based methods such as the Solid Isotropic Material with Penalization (SIMP) approach, the dominant computational cost arises from the repeated solution of large finite element systems at every design iteration. This burden becomes especially severe at engineering-relevant resolutions and in three dimensions, limiting the practical use of high-fidelity topology optimization in design exploration and repeated analysis.

This dissertation develops a runtime-verified multifidelity framework for minimum-compliance topology optimization under a global volume constraint in linear elasticity. The central contribution is a Multifidelity Acceptance Safeguard (MFAS) that periodically evaluates coarse-mesh candidate designs on the fine mesh during the optimization loop. Candidate updates are accepted only if they satisfy the safeguard; otherwise, they are rejected and the coarse state is resynchronized from the last accepted fine-mesh design. By embedding verification within the runtime loop rather than applying it only after optimization, the framework limits the accumulation of coarse-model error while preserving fine-mesh reliability.

Within this verified framework, machine learning is incorporated as an optional warm-start mechanism rather than a replacement for physics-based optimization. A convolutional encoder-decoder (U-Net) predicts coarse-mesh initializations from geometry and boundary-condition descriptors, and all learned proposals remain subject to the same acceptance-rejection protocol as coarse-only candidates.

Four method variants are studied: a fine-mesh reference method, an unverified coarse-only method, a verified multifidelity method, and a verified multifidelity method with U-Net warm start. The framework is evaluated on representative two- and three-dimensional benchmark problems, and structured ablation studies examine the influence of verification frequency and acceptance tolerance. Paired-seed experiments and nonparametric tests are used to assess observed differences in compliance and runtime. The results support the thesis that acceleration in topology optimization becomes substantially more trustworthy when it is coupled with explicit in-the-loop verification.

Streszczenie

Optymalizacja topologii konstrukcji ma na celu wyznaczenie optymalnego rozkładu materiału w zadanej przestrzeni projektowej przy uwzględnieniu równań równowagi oraz określonych kryteriów funkcjonalnych. W metodach opartych na gęstości, takich jak podejście SIMP (Solid Isotropic Material with Penalization), dominujący koszt obliczeniowy wynika z konieczności wielokrotnego rozwiązywania dużych układów równań metody elementów skończonych w każdej iteracji projektowej. Obciążenie to znacząco wzrasta przy rozdzielczościach istotnych z punktu widzenia inżynierskiego oraz w analizach trójwymiarowych, co ogranicza praktyczne zastosowanie optymalizacji topologii opartej na modelach o wysokiej rozdzielczości w eksploracji przestrzeni projektowej oraz analizach powtarzalnych.

Niniejsza rozprawa przedstawia wielopoziomowe podejście do optymalizacji topologii minimalizującej podatność przy globalnym ograniczeniu objętości w ramach liniowej teorii sprężystości, z weryfikacją prowadzoną w trakcie obliczeń. Główną strategią decyzyjną jest mechanizm zabezpieczający akceptację wielopoziomową (MFAS, Multifidelity Acceptance Safeguard), który okresowo ocenia proponowane rozwiązania uzyskane na siatce zgrubnej poprzez ich weryfikację na siatce dokładnej w trakcie procesu optymalizacji. Aktualizacje projektu są akceptowane wyłącznie wtedy, gdy spełniają kryteria tego mechanizmu; w przeciwnym razie są odrzucane, a stan modelu zgrubnego jest ponownie synchronizowany z ostatnim zaakceptowanym rozwiązaniem w oparciu o siatkę dokładną. Dzięki wbudowaniu procedury weryfikacyjnej bezpośrednio w pętlę obliczeniową, zamiast stosowania jej dopiero po zakończeniu optymalizacji, zaproponowane podejście ogranicza kumulację błędów modelu zgrubnego przy jednoczesnym zachowaniu wiarygodności wyników uzyskiwanych na siatce dokładnej.

W ramach zweryfikowanego podejścia wielopoziomowego uczenie maszynowe zostało włączone jako opcjonalny mechanizm inicjalizacji (warm-start), a nie jako zamiennik optymalizacji opartej na modelach fizycznych. Sieć konwolucyjna typu enkoder–dekoder (U-Net) generuje wstępne rozwiązania na siatce zgrubnej na podstawie geometrii oraz warunków brzegowych, przy czym wszystkie propozycje wygenerowane przez model podlegają temu samemu protokołowi akceptacji–odrzucenia podobnie jak rozwiązania uzyskane wyłącznie metodami zgrubnymi.

Przeanalizowano cztery warianty: metodę referencyjną na siatce dokładnej, niew-

eryfikowaną metodę wykorzystującą wyłącznie siatkę zgrubną, zweryfikowaną metodę wielopoziomową oraz zweryfikowaną metodę wielopoziomową z inicjalizacją typu U-Net (warm-start). Zaproponowane podejście oceniono na reprezentatywnych problemach testowych w dwóch i trzech wymiarach, a ukierunkowane badania ablacyjne pozwoliły na zbadanie wpływu częstotliwości weryfikacji oraz tolerancji akceptacji. Do oceny obserwowanych różnic w podatności oraz czasie obliczeń wykorzystano eksperymenty z parami identycznych ziaren losowych (paired-seed) oraz testy nieparametryczne. Uzyskane wyniki potwierdzają tezę, że przyspieszenie w optymalizacji topologii staje się istotnie bardziej wiarygodne, gdy jest powiązane z jawną weryfikacją realizowaną bezpośrednio w pętli obliczeniowej.

List of Figures

List of Tables

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
TO	Topology optimization
FEA	Finite element analysis
DOF	Degree(s) of freedom
SIMP	Solid isotropic material with penalization
OC	Optimality criteria
MMA	Method of moving asymptotes
COC	Continuum-based optimality criteria
CONLIN	Convex linearization
MFAS	Multifidelity acceptance safeguard
PCG	Preconditioned conjugate gradient
CPU	Central processing unit
GPU	Graphics processing unit
Q4	Four-node quadrilateral element
H8	Eight-node hexahedral element
MBB	Messerschmitt–Bölkow–Blohm (beam benchmark)
CV	Coefficient of variation
U-Net	U-shaped convolutional neural network

Notations

Ω	Design domain
N	Total number of finite elements in the bounding-box discretization
N_{act}	Number of active (designable) elements (problems with passive regions)
e	Element index
ρ_e	Element density design variable, $\rho_e \in [0,1]$ (dimensionless relative density)
$\rho_{\min}$	Lower bound on design variables used for numerical stability
$\tilde{\rho}_e$	Filtered density (output of density filter; $\tilde{\rho}_e \in [0,1]$)
$\bar{\rho}_e$	Physical density (output of projection step; passed to the FE solver; $\bar{\rho}_e \in [0,1]$)
$\boldsymbol{\rho}$	Vector of all element design variables
v_f	Prescribed volume fraction
v_e	Element volume (units of length^d, $d = 2$ or $d = 3$)
$V(\bar{\boldsymbol{\rho}})$	Total material volume, $V = \sum_e v_e \bar{\rho}_e$
V_0	Total design-domain volume
$\mathbf{K}(\bar{\boldsymbol{\rho}})$	Global stiffness matrix assembled from element contributions
$\mathbf{K}_e(\bar{\rho}_e)$	Element stiffness matrix
$\mathbf{k}_0$	Unit-modulus element stiffness matrix (reference element stiffness)
$\mathbf{u}$	Global nodal displacement vector
$\mathbf{u}_e$	Element displacement vector extracted from $\mathbf{u}$
$\mathbf{F}$	Global nodal load vector
$C(\bar{\boldsymbol{\rho}})$	Structural compliance objective
E_0	Young's modulus of solid material
$E_e(\bar{\rho}_e)$	Interpolated element Young's modulus (SIMP power law)
$E_{\min}$	Void modulus in SIMP; small fraction of E_0 added to prevent singularity of $\mathbf{K}$
p	SIMP penalization exponent (typically $p = 3$)
ν	Poisson's ratio
$r_{\min}$	Density-filter radius (in element-length units)
$r_{\min,f}$, $r_{\min,c}$	Filter radius on the fine / coarse mesh
$\mathcal{N}_e$	Neighborhood of element e within filter radius $r_{\min}$
H_{ei}	Density-filter weight between elements e and i
$\mathbf{H}$	Sparse matrix of filter weights H_{ei}

$\mathbf{H}_s$	Row-sum normalization vector, $(\mathbf{H}_s)_e = \sum_{i \in \mathcal{N}_e} H_{ei}$
$\mathcal{F}_{r_{\min}}(\cdot)$	Density filter operator (maps $\boldsymbol{\rho} \mapsto \tilde{\boldsymbol{\rho}}$)
$\oslash$	Element-wise (Hadamard) division
$\mathcal{H}_\beta(\cdot)$	Smoothed Heaviside projection operator (maps $\tilde{\boldsymbol{\rho}} \mapsto \bar{\boldsymbol{\rho}}$)
n_x^f, n_y^f, n_z^f	Number of fine-mesh elements in the x-, y-, z-directions
n_x^c, n_y^c, n_z^c	Number of coarse-mesh elements in the x-, y-, z-directions
N_f, N_c	Total number of fine / coarse elements
P	Prolongation operator (coarse $\rightarrow$ fine mesh transfer)
R	Restriction operator (fine $\rightarrow$ coarse mesh transfer)
f_{verify}	Verification frequency (fine-mesh check every f_{verify} coarse iterations)
C_f	Fine-mesh compliance at a verification checkpoint
C_{trial}	Fine-mesh compliance of the current prolongated trial design
C_{prev}	Fine-mesh compliance of the previously accepted design
C_{best}	Best (lowest) verified fine-mesh compliance recorded so far
$C_{\text{coarse,old}}$	Coarse-mesh compliance at the previous verification instant
$C_{\text{coarse,trial}}$	Coarse-mesh compliance at the current verification instant
$\mathbf{x}_m$, $f(\mathbf{x}_m)$	Design point and objective value at iteration m
$\hat{y}(\mathbf{x}_m)$	Surrogate (low-fidelity) prediction at $\mathbf{x}_m$
$\mathbf{x}_i$, $\mathbf{y}_i$	U-Net training pair: input tensor and target density field
κ	Trust ratio adapted to MFAS, $\kappa = \Delta C_{\text{actual}} / \Delta C_{\text{pred}}$
κ_{tr}	Trust-ratio acceptance threshold (2D MFAS)
$\hat{\kappa}$	Empirical MFAS acceptance rate, $\hat{\kappa} = n_{\text{accept}} / (n_{\text{accept}} + n_{\text{reject}})$
δ_{acc}	Improvement margin, $\delta_{\text{acc}} = \sigma_{\text{ratio}} \lvert C_{\text{best}} \rvert$
σ_{ratio}	Improvement margin ratio (2D MFAS acceptance)
ϵ_{rel}	Relative acceptance tolerance (3D MFAS acceptance criterion)
β_{acc}	Acceptance-threshold parameter used in ablation sweeps
N_{cleanup}	Number of fine-mesh cleanup iterations (M3 and M4 variant)
λ	Lagrange multiplier for the volume constraint in the OC update
m	OC move limit
k	Iteration counter
$\text{tol}_{\text{change}}$	Convergence tolerance on density change
k_{consec}	Consecutive-iteration window for the convergence test
$k_{\min}$	Minimum iteration count before convergence is checked
n_{stag}	Early-stopping patience (consecutive non-improving verifications)
t_{gray}	Threshold defining the intermediate-density (gray) band
τ	Visualization threshold for topology binarization
t	Wall-clock time

Chapter 1

Introduction

Structural topology optimization addresses a central question in computational mechanics: given a design domain, boundary conditions, and a performance objective, how should the material be distributed to obtain the best structural response. Sizing and shape optimization address this problem within a fixed structural configuration by adjusting member cross-sections or boundary profiles, respectively, but neither method alters the connectivity of the design. In contrast to sizing and shape optimization, topology optimization allows the connectivity of the structure to change, enabling the discovery of non-intuitive load paths and lightweight designs. Its influence spans conceptual design, compliant mechanisms, and architected materials and is now a well-established aspect of engineering design optimization studies [1, 2].

From a mathematical perspective, topology optimization is commonly posed as a partial differential equation (PDE)-constrained optimization problem. The state variables satisfy the governing partial differential equations, whereas the design variables represent the spatial material distribution in the design domain. In the classical minimum-compliance setting, the objective is to minimize compliance (or maximize stiffness) under a prescribed volume fraction constraint, subject to equilibrium. Foundational work established the continuum viewpoint of the optimal distribution of materials and motivated algorithmic developments that are still widely used today [3, 4].

Despite the maturity of the field, topology optimization remains computationally demanding at high resolutions, particularly for three-dimensional problems. Standard density-based solvers may require hundreds of iterations, and each iteration requires at least one finite element analysis of a large, sparse system. The computational cost scales with the number of degrees of freedom, load cases, and the solver tolerance. Consequently, high-fidelity optimization can impede interactive design exploration, large parametric studies, and statistically rigorous evaluations of algorithmic variants [5]. This dissertation develops a framework that explicitly accounts for this computational reality while preserving trust in the final design.

1.1 Background and Motivation

Topology optimization can be formulated through distinct paradigms—including density-based, level-set, phase-field, and evolutionary approaches, each with characteristic computational properties and domain representations (see Chapter 2, Section 2.1.1 for a brief review on these methods). This dissertation adopts the density-based framework, specifically the Solid Isotropic Material with Penalization (SIMP) method [4, 6], for three reasons: (i) SIMP provides a well-established baseline with known convergence properties and numerical behavior, enabling rigorous benchmarking of the proposed acceleration strategy; (ii) the method's reliance on repeated finite element analysis at each iteration creates a clear computational bottleneck that multifidelity strategies are well-suited to address; and (iii) the continuous density representation facilitates the inter-mesh transfer operations—restriction and prolongation—that are central to the proposed verification framework. Among density-based formulations, SIMP is widely adopted because it integrates naturally with standard finite element discretizations and supports efficient gradient-based updates through adjoint sensitivity analysis.

However, straightforward density-based formulations exhibit well-documented numerical and modeling challenges. Numerical instabilities include checkerboard patterns, mesh-dependent feature sizes, and sensitivity to local minima [7]. Therefore, filtering and projection schemes are widely used to regularize the problem, impose a length scale, and obtain physically meaningful and manufacturable designs [8]. These techniques substantially improve the numerical behavior, but they do not eliminate the dominant computational burden of repeated large-scale partial differential equation (PDE) solutions. In this dissertation, "PDE-constrained" refers to enforcing the governing *linear elasticity boundary-value problem*—posed as a partial differential equation in the continuous design domain Ω—through its finite-element discretization at each design iteration [9, 10]. The methodology and mathematical formulations are detailed in Chapter 3.

The computational challenge becomes increasingly pronounced for fine meshes and three-dimensional problems. The cost of assembling the stiffness matrix, applying filtering operations, and solving equilibrium equations can dominate the computational runtime. The burden grows further with multiple load cases and repeated runs across the parameter choices (filter radius, continuation schedules, and penalization). As emphasized in recent studies on topology optimization, acceleration remains essential for broadening the practical reach of topology optimization from offline computation to more interactive design workflows [5].

Two acceleration avenues are particularly relevant to this study. The first is *multifidelity* optimization, where cheaper models (e.g., coarser meshes or reduced-order approximations) guide most of the iterative search, whereas expensive high-fidelity

evaluations are used selectively [11]. The second is *machine learning* (ML), which has been explored for predicting near-optimal designs, learning update operators, and producing strong initial designs that reduce the number of expensive optimization iterations [12, 13, 14]. In both instances, the fundamental objective remains consistent: to decrease the computation time by reallocating certain tasks away from repeated fine-mesh PDE solves.

These acceleration strategies share a common limitation: they may reduce reliability in pursuit of computational efficiency. A design that appears adequate on a coarse mesh may not perform well when evaluated on finer meshes. Surrogates have the potential to introduce bias, which can divert the optimization path from reaching high-quality solutions. Machine learning predictors can propose design topologies that appear feasible but fail to accommodate new boundary conditions, load scenarios, or mesh resolutions [14, 15]. In engineering practice, these risks are problematic because the cost of a poor design may far exceed the computational savings achieved. Therefore, this dissertation adopts a reliability-oriented viewpoint: acceleration should be paired with verification so that efficiency gains are achieved without sacrificing confidence in the solution quality.

1.2 Problem Statement

This dissertation focuses on minimum-compliance topology optimization under a global volume constraint in linear elasticity. A conventional fine-mesh SIMP solver with standard regularization serves as the baseline reference [6, 7]. The problem is to reduce the runtime of this baseline while ensuring that the accelerated method produces designs of comparable quality.

A purely coarse mesh strategy can reduce the per-iteration cost but may converge to topologies that do not transfer well to fine meshes. Similarly, surrogate models and ML predictors can propose designs in less time, but their accuracy can deteriorate outside the training distribution or at the late stages of optimization, where small compliance differences are meaningful [14, 15]. Therefore, the key technical difficulty is reliability, that is, how to use cheaper coarse models and data-driven proposals while maintaining the desired quality of the final topology design.

To address this, a verification-driven multifidelity strategy is developed. Most iterations are performed on a cheaper (coarse) model to generate candidate-design updates. At prescribed intervals, these candidates are evaluated using a trusted high-fidelity model (typically a fine mesh). An acceptance criterion determines whether a candidate is sufficiently reliable to propagate. If the candidate fails verification, it is rejected, and the method reverts to the last verified design. An acceptance tolerance criterion is introduced to balance early-stage exploratory progress against late-stage

accuracy requirements. Therefore, verification is incorporated within the runtime loop rather than being confined to post-processing diagnostics.

Machine learning is integrated while ensuring that concerns related to reliability and solution quality are addressed. ML components are used as optional accelerators, primarily by producing informed initializations and candidate updates. Their outputs are not assumed to be correct a priori; instead, the same high-fidelity verification mechanism functions as a safeguard to prevent poor generalization from adversely affecting the final design process. In this framework, ML becomes a tool for reducing computational effort when it helps, while verification ensures that the final result remains competitive with trusted baselines.

Throughout this dissertation, M3 (MFAS without ML) is the primary validated contribution. It provides the most favorable overall verified speedup–compliance trade-off across the benchmark set. M4 (MFAS with U-Net initialization) is the ML-assisted extension: it investigates whether a learned warm start can further improve initialization quality and acceptance behavior within the same acceptance-safeguarded loop. M4 builds directly on the verified optimization infrastructure established by M3 and is evaluated under the same acceptance criteria. The objective is not to demonstrate universal ML-driven acceleration, but to determine under what conditions ML-assisted initialization yields measurable benefit within a verification-constrained workflow.

1.3 Research Objectives

This study was guided by four objectives:

Objective 1: Achieve verified computational acceleration relative to a fine-mesh SIMP baseline. Develop a runtime-verified multifidelity topology optimization framework that targets a reduced computational cost relative to a fine-mesh SIMP baseline while preserving a comparable final compliance.

Objective 2: Ensure reliability through an acceptance/rejection mechanism that prevents silent solution degradation. Formulate and implement an acceptance/rejection (with recovery) mechanism that limits the accumulation of coarse-model and surrogate-induced errors, thereby preventing silent degradation of the solution quality.

Objective 3: Demonstrate robustness of the proposed framework across the SIMP-based benchmark variants considered in this study. Demonstrate that the proposed framework yields consistent speed–quality performance across the SIMP-based benchmark variants considered in this study.

Objective 4: Evaluate the incremental effect of ML-assisted initialization within the verified multifidelity workflow. Integrate a U-Net warm-start initializer within the verification-guided multifidelity loop and measure its effect on initialization

quality, acceptance behavior, and wall-clock time, while assessing whether the acceptance mechanism bounds the reliability risk introduced by the learned component.

1.4 Research Questions

The objectives led to the following research questions:

RQ1 (Acceleration): Can a runtime-verified multifidelity strategy reduce the computational cost relative to a fine-mesh SIMP baseline while maintaining a comparable final compliance?

RQ2 (Reliability): Does the acceptance mechanism prevent the accumulation of coarse-model and surrogate-induced errors, thereby avoiding the silent solution degradation that may occur in coarse-only or surrogate-only approaches?

RQ3 (Robustness): Is the framework robust across the SIMP-based benchmark variants considered in this study?

RQ4 (Machine learning contribution): When ML-assisted initialization is integrated into the verified multifidelity workflow, does it improve initialization quality, acceptance behavior, or wall-clock time without compromising reliability, and under what verification-schedule conditions do these improvements translate into end-to-end runtime savings?

1.5 Contributions

This dissertation makes three contributions to the literature.

(i) A runtime-verified multifidelity optimization framework for compliance-based topology optimization. The primary contribution of this study is a runtime-verified multifidelity strategy for PDE-constrained topology optimization, which is benchmarked against a fine-mesh SIMP baseline [6, 16]. Coarse model iterations are treated as proposal steps, and periodic high-fidelity evaluations are used as acceptance tests. The framework includes (i) an explicit acceptance/rejection rule based on a high-fidelity compliance evaluation, (ii) an acceptance tolerance criterion intended to balance computational savings against solution reliability over the course of optimization, and (iii) a rejection-and-recovery mechanism that reverts to the last verified state when a proposal is not accepted.

(ii) Verification-guided integration and assessment of ML-assisted initialization. The second contribution is the integration of a U-Net warm-start initializer as an optional accelerator within the verified multifidelity loop. Its output is not assumed to be correct; it is treated as a coarse-mesh starting point and is subject to the same periodic high-fidelity verification applied to every other coarse proposal. This design

allows the effect of the learned initializer on initialization quality, acceptance behavior, and runtime to be isolated and measured empirically rather than assumed.

(iii) Benchmark-based validation on SIMP-derived variants emphasizing the speed–quality–reliability trade-off. This dissertation provides an empirical validation of the SIMP-based benchmark variants considered in this study, reporting the computational efficiency and quality preservation relative to the fine-mesh references. The analysis highlights the practical trade-off between acceleration and reliability, which is consistent with the observations in surveys and discussions of large-scale topology optimization [1, 14, 15].

Parts of the benchmark study reported in this dissertation have been published in journal form as Tatke and Kaczmarczyk [17]. The thesis extends that publication through a fuller methodological exposition, expanded ablation analysis, implementation details, and the complete dissertation context.

1.6 Scope and Limitations

The scope of this dissertation is minimum-compliance topology optimization in linear elasticity with a global volume constraint, using density-based (SIMP) interpolation and standard regularization techniques [4, 7, 18]. The focus is on accelerating the optimization workflow through multifidelity based approach and ML as initializer.

Several extensions are intentionally outside the scope of this study. Geometric nonlinearity, nonlinear material behavior, and contact were excluded [19, 20]. Multiphysics optimization, including thermal or coupled thermomechanical problems [21, 22], is not included in the current thesis version and is considered a future research direction rather than a case study. These boundaries were chosen to isolate the methodological question addressed here: how to combine acceleration mechanisms with runtime verification in PDE-constrained topology optimization while maintaining confidence in the solution quality.

1.7 Thesis Organization

Chapter 1 introduces the research problem, motivation, and objectives. It establishes the context for multifidelity topology optimization and outlines the novel contributions of the proposed framework.

Chapter 2 provides a comprehensive review of the literature on density-based topology optimization, multifidelity strategies, and recent machine learning approaches. This review motivates the need for verification mechanisms that support acceleration without sacrificing reliability and positions the proposed framework within the current state of the art.

Chapter 3 presents the mathematical formulation and algorithmic framework, including compliance minimization, sensitivity analysis, regularization, the neural network architecture used, and, the proposed multifidelity with acceptance safeguard mechanism.
Chapter 4 outlines the experimental methodology and the implementation plan that include definition of benchmark, a mesh hierarchy to be employed in the experimentation, the parameters, the flow of execution and the performance metrics to be utilized for comparing all method variants.
Chapter 5 reports and analyzes the results, which includes quality/speed trade-off, ablation study on verification schedules and tolerance parameters, and scalability observation with respect to the 2D and 3D benchmarks.
Chapter 6 concludes the dissertation by summarizing the contributions, answering the research questions, and outlining future work.

Chapter 2

Literature Review

2.1 Topology Optimization Fundamentals

2.1.1 Historical Development

Topology optimization began as a method to overcome the limitations of sizing and shape optimization, because both methods improve the member size and/or shape within the prescribed configuration. Engineering systems requiring optimized "load path" configurations and void placement necessitated methods to optimize structural connectivity through material distribution in a reference domain. The first major development employed relaxation concepts based on homogenization theory, linking spatial variability of effective properties to parameterized microstructures, enabling minimum compliance formulations with global volume constraints and mathematically consistent interpretation of intermediate densities [3]. *Density-based formulations* emerged as practical approaches, reformulating optimal shape design as material distribution problems using continuous density functions rather than traditional boundary variation techniques [4]. Continuum-based optimality criteria algorithms simultaneously optimized topology and geometry through staged processes [23].

Level-set methods introduced implicit boundary representations through higher-dimensional level-set functions, tracking structural boundary evolution with sharper interfaces than density-based approaches [24, 25]. Advanced level-set implementations incorporated parametric formulations enabling efficient gradient-based optimization [26, 27]. *Phase-field methods* have also been employed in topology optimization studies. Here, the topology optimization problem is formulated as a continuous problem with the phase-field as design variables within a fixed reference domain [28, 29]. *Evolutionary methods* were developed as alternative paradigms where low-stressed material is progressively eliminated through iterative finite element analysis, offering visualization of evolutionary optimization paths [30]. These methods were later extended using metaheuristic techniques including genetic algorithms, particle swarm optimization, artificial bee

colony, and differential evolution for truss and structural optimization [31, 32, 33, 34, 35, 36].

Topology optimization expanded beyond stiffness to function-driven synthesis, including compliant mechanisms [18]. Modern implementations leverage parallel computing and sensitivity filtering [37]. These developments established compliance minimization as canonical testbeds, with density-based methods remaining central due to their computationally tractable framework for discovering structural connectivity under PDE constraints. Table 2.1 provides a concise comparison of classical topology optimization methods, placing SIMP within the broader landscape of density-based, level-set, and evolutionary formulations.

Table 2.1: Comparison of classical topology optimization methods

Method	**Design Representation**	**Key Strengths**	**Main Limitations**
Density-based (SIMP) [1, 2, 4]	Element-wise continuous density field	Efficient; simple FE coupling; widely adopted	Grey regions; needs penalization/filtering
Level-set methods [24, 25, 38]	Implicit boundary via level-set function	Sharp boundaries; clear topology evolution	Reinitialization/ remeshing; higher cost
Phase-field methods [28, 29]	Diffuse interface via phase-field variable with PDE regularization	Mesh-independent regularization; smooth topology changes; clear variational structure	Additional PDE solve(s); parameter tuning (interface width); thresholding/post-processing often needed
Evolutionary (ESO) [30, 36]	Discrete removal/addition of elements	Intuitive; discrete designs	Heuristic; weaker convergence guarantees

2.1.2 Solid Isotropic Material with Penalization (SIMP)

The SIMP method originated from Bendsøe's reformulation of optimal shape design as a material distribution problem [4]. This approach introduced the use of a continuous density function as the design variable, where regions of high density define the structural boundaries and intermediate densities are governed by an artificial material law. The fundamental principle is to assign each finite element a density variable $\rho_e \in [0, 1]$ and interpolate the element stiffness through a power-law relationship with penalization exponent p.

In SIMP formulations, a penalization exponent (typically $p = 3$) is used. It assigns intermediate-density elements disproportionately low stiffness relative to the material they consume, thereby economically penalizing "gray" solutions and driving the

optimizer toward binary (solid-void) topologies [39]. Modern implementations adopt a *modified SIMP* formulation that includes a small void modulus $E_{\min}$ (an ersatz Young's modulus assigned to void elements) to prevent singularity of the global stiffness matrix, ensuring that the finite element equilibrium equations remain solvable at all iterations [16]. This modification differs from the classical SIMP approach, which instead impose a lower bound slightly above zero on the density variables themselves.

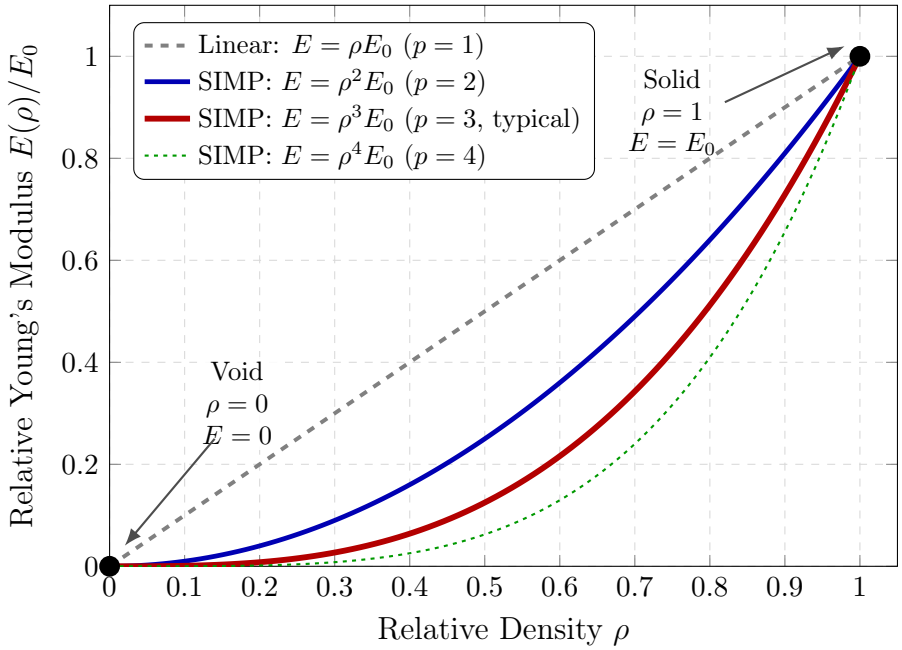

Figure 2.1: SIMP material interpolation curves showing variation of relative Young's modulus $E(\rho)/E_0$ with relative density ρ for different penalization factor p.

Figure 2.1 illustrates how different penalization factors affect the stiffness-density relationship, showing that higher values of p more aggressively suppress intermediate densities. The computational tractability of the SIMP approach which requires only element-wise density updates and standard finite element analysis – combined with its compatibility with gradient-based optimizers and the ability to handle general boundary conditions and loading scenarios, has established it as the de facto standard for density-based topology optimization [16, 39].

SIMP serves as the baseline approach used throughout this thesis. The complete mathematical formulation, including the density filter pipeline and the specific interpolation equation employed in the implementation, is detailed in Chapter 3, Section 3.2.1.

2.1.3 Gradient-Based Methods for Topology Optimization

Gradient-based optimization methods form the computational backbone of modern topology optimization, with Optimality Criteria (OC) and the Method of Moving Asymptotes (MMA) emerging as the two dominant approaches in structural design applications.

OC methods, popularized through the pioneering continuum-type optimality criteria (COC) algorithm [23], have become particularly prevalent in SIMP-based topology optimization due to their computational efficiency and straightforward implementation. These methods derive update schemes directly from the Karush-Kuhn-Tucker (KKT) optimality conditions, resulting in explicit, closed-form expressions for design variable updates. The primary advantage lies in their minimal computational overhead—requiring no matrix inversions or complex subproblem solutions—making them exceptionally cost-effective for problems with simple constraint structures, particularly volume constraints in compliance minimization. However, OC methods exhibit limitations in generality, particularly with multiple constraints and complex problem formulations [2], [40].

MMA, introduced by Svanberg [41], addresses OC's limitations through sequential convex approximations of the optimization problem. This approach provides robust convergence across diverse constraint types and has become widely adopted for complex, multiply-constrained topology optimization problems. MMA's generality enables extension to stress constraints, displacement constraints, and multiphysics applications, though at moderately higher computational cost per iteration compared to OC [40, 42].

Both methods rely on accurate sensitivity information, obtained through sensitivity analysis within the finite element framework. In practice, sensitivities are computed efficiently using the adjoint approach, in which one additional adjoint solve per scalar response yields gradients with cost largely independent of the number of design variables. The adjoint method enables efficient computation of objective and constraint sensitivities with computational cost essentially independent of the number of design variables—a crucial property for large-scale problems with thousands of elements. The choice between OC and MMA reflects a trade-off between computational efficiency and problem generality, with OC preferred for simple compliance problems while MMA serves as the robust choice for applications requiring multiple constraint handling [10].

2.1.4 Filtering and Regularization

Topology optimization has become a mature design tool that can be applied to many industrial problems and coupled-field design challenges; however, considerable numerical instabilities are still encountered and must be managed through filtering and regularization techniques [7].

The numerical issues in topology optimization fundamentally arise from the ill-posed nature of the original problem. Unless the design space is constrained, mesh refinement can drive the solution toward ever-finer perforations—an indication of non-convergence and yielding mesh-dependent optima (i.e., qualitative differences in the optimal design obtained from different discretization meshes). Additionally, checkerboard patterns

may emerge as regions of alternating solid and void elements that can occur in optimal topologies but are, in practice, artifacts of incorrect finite-element modeling rather than true features of the optimum. Both types of numerical instability necessitate the use of regularization and filtering techniques that facilitate the existence of well-defined solutions and promote convergence of the finite-element approximation [7, 43].

Several approaches can be used to address the aforementioned numerical stability issues. The mesh-independent filtering technique developed by Sigmund [7] addresses mesh dependency by modifying the design sensitivities through a weighted average of sensitivities in the neighborhood of each element. Although this approach is heuristic, solutions similar to those produced by gradient-constrained approaches are often obtained. Very little additional computation is required, and the complexity of implementation is substantially reduced when compared with other approaches.

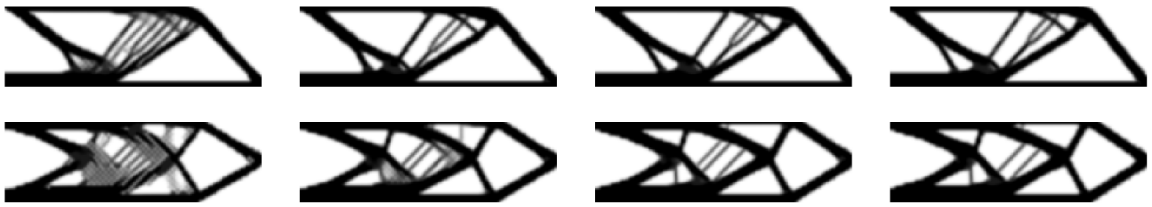

Figure 2.2: Representative SIMP density-field snapshots (two benchmark problems) illustrating the qualitative effect of filtering and regularization: fine-scale numerical artifacts and gray regions are progressively suppressed, yielding clearer structural members and a controlled length scale.

The filtered, penalized artificial material approach [43] describes a continuous density field over the domain that exists at every point in the domain and varies smoothly between solid (material) and void (no material). The corresponding material properties are then interpolated nonlinearly in terms of this density. Convolution-based filtering techniques use regularized density fields to replace the direct dependence of material properties on the pointwise density. By regularizing the density field, sharp changes in material properties that are associated with ill-posedness are suppressed. The Heaviside projection method [8] is a further advanced regularization technique in which smoothed Heaviside functions are used to project the design-variable field. Control of the minimum length scale of design features can also be achieved, resulting in nearly discrete, manufacturable designs without requiring additional design constraints. Figure 2.2 illustrates representative density-field snapshots showing how filtering and regularization suppress checkerboarding and reduce intermediate (gray) regions during SIMP optimization.

The previously established filtering and regularization methods have also been extended to the nonlinear regime to address complications associated with nonlinear elastic behavior and large deformation [20]. In such formulations, filtering methods are

integrated to smooth both the design variables and the associated sensitivities within the nonlinear finite-element framework and sensitivity analysis, while effectiveness is maintained despite additional complications due to nonlinear material behavior and geometric effects. Filtering and regularization techniques are now considered necessary for robust topology optimization practice and have been used to address checkerboarding and mesh dependency, thereby producing convergent and manufacturable designs across a variety of applications.

2.1.5 Classical Surrogate-Model-Based Optimization

Classical surrogate-model-based optimization aims to reduce the repeated evaluation of high-fidelity simulation models (often governed by PDEs) that occur during design optimization and feasibility searches [44]. To accomplish this goal, surrogate models are constructed to be substantially less computationally intensive than the underlying analysis models while approximating the relevant input-output relationships of the physics with acceptable accuracy over a defined region of the design space. In engineering design applications, such substitutes can significantly reduce the number of finite element or computational fluid dynamics (CFD) analyses required to guide the search for improved designs [45, 46, 47].

One of the earliest and most widely used surrogate paradigms is the response surface methodology (RSM). In RSM, the design space is sampled at a set of points, and low-order polynomial or regression models are fitted to approximate the objective and constraint responses [44, 48]. These models are attractive because of their simplicity, rapid evaluation, and analytical differentiability. However, their predictive quality can deteriorate quickly when the response exhibits strong nonlinearity, sharp transitions, or high-dimensional dependence, which are features common in topology optimization and other PDE-constrained problems [45].

Reduced-order models (ROMs) provide a more physics-informed surrogate. Here, the governing equations are projected onto a low-dimensional subspace such that the dominant deformation, energy, and/or flow features are retained, whereas less influential components are truncated. When the reduced basis accurately captures the relevant solution manifold, ROMs can yield significant speedups while remaining consistent with the underlying mechanics. A key limitation is that ROM accuracy is often reliable only within a restricted region of the parameter, geometry, or loading space, and constructing a robust reduced basis may require substantial offline preprocessing [45, 48].

Probabilistic surrogates, including Kriging and Gaussian process regression, offer additional flexibility by modeling both the predicted response and an associated uncertainty. This uncertainty information can be used to guide adaptive sampling and balance exploration and exploitation [48]. Extensions, such as gradient-enhanced

Kriging, further improve data efficiency when sensitivity information is available [49]. Nevertheless, classical surrogate workflows are typically trained offline and then treated as fixed during optimization [44]. Moreover, many traditional frameworks do not include systematic runtime verification against high-fidelity models; thus, surrogate errors may accumulate as the optimization trajectory leaves the region where the surrogate is valid, potentially resulting in suboptimal or infeasible designs [47, 48].

2.1.6 Implementation Codes in Topology Optimization Literature

Topology optimization research has been significantly enhanced through openly accessible implementation codes that have enabled researchers and students to understand and apply these methods. The 99-line MATLAB code by Sigmund [6] provided a compact implementation of an optimizer, mesh-independent filter, and finite element analysis in an extremely concise format. Sigmund's work laid the foundation for numerous extensions: Andreassen et al. [16] developed an 88-line code with improved computational efficiency using vectorized assembly and filtering strategies, while Ferrari et al. [50] introduced top99neo and top3D125, achieving significant speed-ups for 2D problems and significant improvements for 3D implementations through optimized matrix assembly and filter operations. Liu et al. [51] provided a 169-line MATLAB code for 3D problems with systematic approaches for boundary conditions and loading definitions.

In addition to MATLAB, the topology optimization community has utilized diverse programming environments to democratize access. Python-based implementations have also gained popularity, with Zuo et al. [52] developing a 100-line code integrated with Abaqus for the BESO methodology, and Gupta et al. [53] creating a 55-line FEniCS implementation supporting parallel computation on multi-core systems. The FEniCS ecosystem facilitates multiphysics problems through symbolic specification of variational forms, as demonstrated by Ferro et al. [54] and extended by Yan et al. [55] through the ATOmiCS toolbox, which couples FEniCS with the OpenMDAO framework for automated derivative computation using the adjoint method.

Wang et al. [56] provided a comprehensive review classifying open-source codes based on methodology (SIMP, BESO, etc), programming environment, and complexity level, documenting over 30 different implementations for both research and educational purposes. Collectively, these codes bridge the gap between theoretical formulations and practical applications, enabling rapid prototyping and benchmarking while maintaining pedagogical clarity.

2.2 Multifidelity Methods

2.2.1 General Multifidelity and Mesh Refinement

Multifidelity optimization employs models of differing resolution, computational cost, and accuracy to improve time-to-solution at comparable predictive quality. For structural topology optimization, the computational expense originates primarily from repeatedly solving large, sparse linear systems that arise from finite element discretizations. Therefore, multifidelity optimizations commonly use mesh-based hierarchical fidelities, where the low-fidelity (LF) model corresponds to a coarser discretization with fewer degrees of freedom, and the high-fidelity (HF) model corresponds to a finer discretization serving as the accuracy reference [5]. The use of mesh-based hierarchies creates significant numerical challenges: design variables (typically represented by the element-wise density field $\rho \in [0, 1]$) must be mapped between meshes, and the compliance and sensitivity information must remain consistent across resolutions as well as with regularization techniques (such as filtering and length-scale control) used to obtain mesh-independent solutions.

One popular approach follows a coarse-to-fine methodology, where early iterations use the LF mesh to explore candidate layouts, and later iterations use the HF mesh to refine promising designs. This strategy can substantially reduce computational cost through a one-way transition after a prescribed coarse stage (e.g., after a fixed number of iterations or coarse convergence) [57, 58]. Multilevel and multigrid concepts extend this idea by serving as both efficient linear solvers and algorithmic accelerators that exploit multiple resolutions within the optimization loop [5, 59]. Restriction and prolongation operators transfer density fields between mesh levels while maintaining consistency across resolutions [59, 60]. However, uniform refinement strategies treat the entire design domain identically, refining all regions regardless of local solution complexity, which motivates adaptive refinement to concentrate resolution where it is most needed [58, 61].

In contrast, h-adaptive methods selectively refine mesh resolution in regions where solution gradients are steep or material boundaries evolve rapidly. Nguyen et al. [62] and Costa et al. [63] employed hanging-node h-refinement integrated with SIMP and r-adaptivity (mesh repositioning), respectively, to track evolving structural boundaries. These spatially adaptive approaches achieve substantial cost reductions—often 50–70% fewer degrees of freedom compared to uniform fine meshes—while preserving solution accuracy. From a multifidelity perspective, the spatial interpretation is that critical regions receive high-fidelity treatment while bulk regions remain low-fidelity. However, h-adaptive methods introduce algorithmic complexity (hanging nodes, dynamic data structures) and require expensive error estimators for refinement decisions.

When viewed from the broader perspective of surrogate-based optimization, cor-

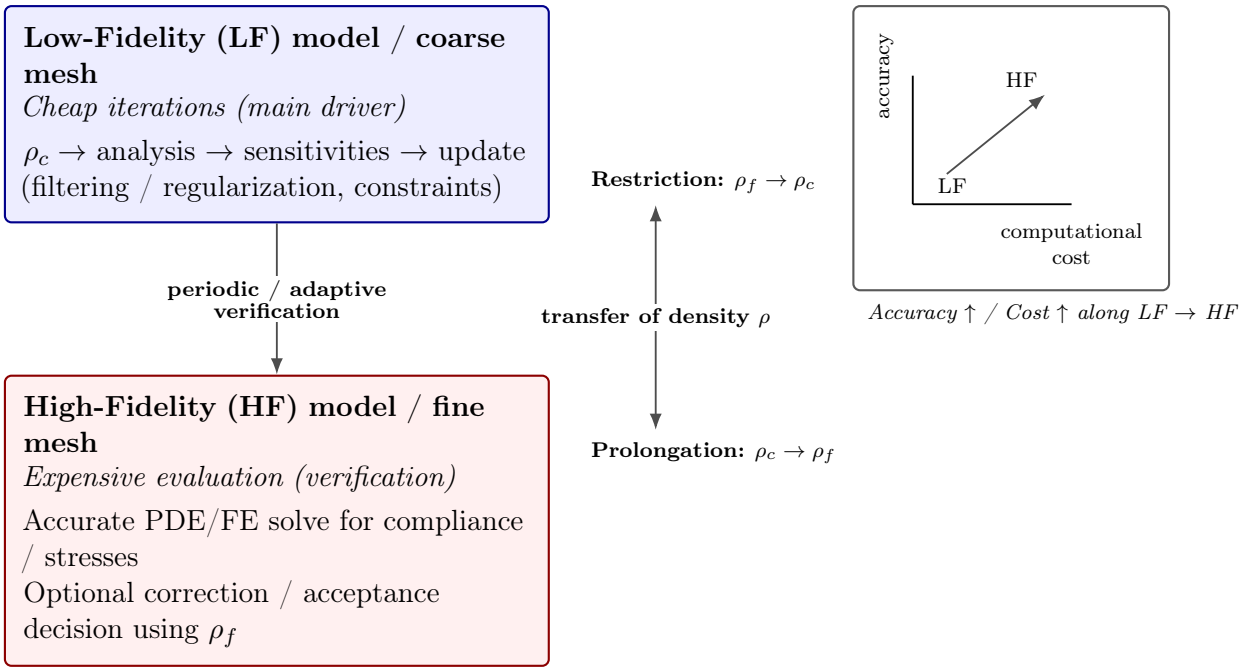

Figure 2.3: Multifidelity optimization hierarchy.

related Gaussian process approximations such as co-kriging enable optimization with multiple analysis levels [64]. Multilevel methods traditionally characterize approximation quality and computational cost through convergence rates, distributing effort accordingly among models [65]. The surrogate-based optimization community has explored multifidelity frameworks using kriging models and particle swarm optimization to address computationally expensive multidisciplinary problems [66]. These approaches are typically applied when function evaluations are expensive but do not naturally integrate into iterative PDE-constrained optimization workflows like topology optimization.

Recently, the topology optimization community has explored data-driven multifidelity design, where solution candidates initially generated by low-fidelity optimization are iteratively updated through variational autoencoders (VAEs) and high-fidelity evaluation [11, 67]. This approach splits the optimization problem into two components: LF topology optimization and HF evaluation, indirectly addressing problems that are analytically or numerically difficult to solve directly. Multifidelity VAE frameworks have been developed to enhance computational efficiency by combining low-fidelity and high-fidelity data while maintaining design quality [68, 69]. Separately, reduced-order modeling equipped with *a posteriori* error bounds provides an additional fidelity dimension, where approximate state solutions estimate the discrepancy between approximate and high-fidelity solutions through rigorous error estimators [70].

Figure 2.3 illustrates the LF–HF hierarchy, which uses periodic high-fidelity verification to enable controlled switching between coarse and fine solvers. Restriction and prolongation operators transfer the density field between mesh resolutions to main-

tain consistency. Existing multifidelity TO methods—whether uniform continuation, h-adaptive, or surrogate-based—lack systematic runtime verification that low-fidelity proposals satisfy the governing PDEs. Fidelity transitions rely on iteration counts (continuation), error estimates requiring expensive dual solves (h-adaptivity), or offline training (data-driven methods). None employ accept/reject logic based on PDE residuals computable from the low-fidelity solution itself. This verification gap motivates the multifidelity framework developed in Chapter 3, which embeds physics-based correctness checks directly into the optimization loop without requiring additional fine-mesh solves or dual problem evaluations.

2.2.2 Trust Region Methods

Trust-region methods provide a systematic framework for managing approximation models in optimization, enabling controlled use of computationally cheaper models while ensuring convergence. Originating in nonlinear programming, these methods now play a central role in multidisciplinary, multifidelity design optimization [71, 72].

The fundamental premise of the trust-region concept is that a neighborhood is established around the current design point within which the surrogate model is considered sufficiently reliable for optimization purposes. Alexandrov et al. [73] formalized this concept for structural optimization, developing an adaptation strategy for the trust-region radius based on how well the surrogate model prediction agreed with the actual objective function value. The classical trust-region framework quantifies this agreement through a ratio comparing actual improvement to predicted improvement. If the ratio indicates strong agreement, the trust region is expanded to allow larger exploratory steps; if the surrogate proves unreliable, the region is contracted to force more conservative updates. At each iteration, the surrogate is minimized within the trust region, the proposed design is accepted or rejected based on sufficient decrease conditions, and the radius is updated until convergence.

In multifidelity applications, trust-region methods enable adaptive switching between models of varying cost and accuracy. Alexandrov et al. [74] demonstrated this capability in aerodynamic optimization with variable-fidelity models spanning empirical correlations, panel methods, and computational fluid dynamics simulations. Their model management framework incorporated correction-based approaches (where low-fidelity models are corrected using sparse high-fidelity data) and direct substitution strategies. Forrester et al. [75] applied trust-region concepts to optimization using partially converged CFD simulations, achieving substantial computational savings by exploiting correlations between convergence levels while maintaining solution quality through rigorous acceptance criteria.

Within PDE-constrained optimization, trust-region strategies have been combined

with model order reduction techniques to achieve significant speedups. Qian et al. [76] developed a certified trust-region reduced basis approach coupling dimensionality reduction with rigorous error bounds, enabling acceptance decisions based on computationally inexpensive error estimators rather than full-order evaluations. Keil et al. [77] extended this to nonconforming dual approaches for parameter optimization, while Kartmann et al. [78] demonstrated adaptive parameter and state space reduction within a trust-region iteratively regularized Gauss-Newton method for inverse problems. These methods construct reduced-order models by projecting the governing equations onto low-dimensional subspaces, then employ trust-region logic to verify when the reduced model provides sufficient accuracy through residual-based error estimators and certified bounds.

Despite these advances, current trust-region frameworks exhibit a critical limitation for topology optimization: trust metrics lack explicit physics validity checks beyond convergence of the optimization objective. The acceptance criteria focus on objective function agreement and error estimator bounds, but do not directly verify whether the underlying partial differential equations remain satisfied to acceptable accuracy. In multifidelity CFD or structural analysis, this gap is mitigated by domain-specific heuristics and the smooth nature of parametric variations. However, topology optimization presents unique challenges due to the discrete nature of material distribution and the potential for low-fidelity models to propose designs that severely violate equilibrium despite appearing promising in terms of objective value. This absence of runtime physics verification represents a research gap that the present work addresses through compliance-based acceptance criteria, as formalized in Chapter 3, Section 3.4.3.

2.2.3 Multifidelity Model Management in Topology Optimization

Multifidelity model management is the process by which multiple fidelity levels are used to control approximation errors and to manage transitions between fidelities throughout the optimization process. While originally developed for multidisciplinary design optimization, it also addresses the basic question of when to use inexpensive approximations and when to use expensive high-fidelity evaluations [73]. In addition to correcting low-fidelity predictions using sparse high-fidelity data via additive or multiplicative corrections, space mapping approaches transform the low-fidelity design space to match the high-fidelity response surface [48]. Examples of successful applications of these approaches include aerodynamic optimization involving empirical correlations, panel methods, and computational fluid dynamics (CFD) simulations [74].

In topology optimization, managing model fidelity presents new challenges due to the high dimensionality of the design variables and the fact that discrete material

distribution decisions create structural connectivity. In current multifidelity topology design frameworks, the workflow is commonly organized around low-fidelity optimization and high-fidelity evaluation, implemented either as two coarse-grained phases or as an interleaved process in which high-fidelity analyses are performed periodically to verify and refine low-fidelity proposals [11]. Low-fidelity optimization utilizes simplified physics or coarse discretizations to quickly generate candidate designs, whereas high-fidelity analysis is used to verify their mechanical performance. In this manner, indirect solution of analytically or numerically difficult problems can be achieved using iterative refinement [79]. Data-driven extensions of these frameworks utilize variational autoencoders to generate candidate solutions from low-fidelity optimizations, which are subsequently evaluated and iteratively updated using high-fidelity models [67].

One major shortcoming of current multifidelity topology optimization frameworks is the lack of systematic runtime verification mechanisms. In current multifidelity model-management practice, transitions between fidelity levels are typically managed using continuation strategies (predefined iteration schedules) [80, 81], h-adaptive methods (expensive dual-problem error estimates) [61, 82, 83, 84], or machine learning approaches (offline training data distributions) [11, 67, 69, 79]. Accept/reject decisions based on governing physics residuals that can be computed directly from low-fidelity solutions are not very commonly in current multifidelity topology optimization frameworks. Here, the term "physics residual" is used to denote a discrete equilibrium residual of the governing PDE after discretization, evaluated using quantities available at the current fidelity level. Such a residual based check is distinguished from objective function agreement, since satisfaction of the governing equations is assessed directly rather than only improvement in the optimization objective. Furthermore, as low-fidelity proposals increasingly fail to satisfy equilibrium constraints, and/or as surrogate model drift accumulates over multiple iterations, the lack of such verification mechanisms increases the risks that optimization may proceed toward infeasible or suboptimal regions without being detected [65, 73, 85].

2.2.4 Verification and Validation of Optimized Solutions Using Acceptance Criteria

Verification and validation of optimized solutions using acceptance criteria are quantitative decision rules used to determine whether approximate model predictions are sufficiently accurate to support the progression of optimization. Trust-region methods define neighborhoods about the current design in which surrogate models are assumed to be reliable and utilize the agreement ratio between the actual and predicted improvements to adaptively modify the size of the trust region [48, 73].

Recent developments in the application of reduced order models within trust re-

gion frameworks for PDE constrained optimization utilize rigorous a-posteriori error estimators. Such methods employ sufficient and necessary optimality conditions to verify trial step acceptance [78]. A sufficient condition for acceptance verifies that the reduced-basis objective plus error bound remains below the previous value, while rejection occurs when the reduced-basis objective minus error bound exceeds this threshold. When cheap error bounds provide no decision, full-order model evaluation resolves acceptance. In case of acceptance, the trust region radius is enlarged if the reduced-basis objective accurately predicts the actual reduction, verified by checking whether the ratio of actual to predicted reduction exceeds a threshold typically set between 3/4 and 1. Thus, this hierarchical verification strategy enables guaranteed convergence while minimizing computational expense to critical iterations. Adaptive hierarchies combining machine-learning surrogates, reduced-basis models, and full-order models enable error-aware sufficient-decrease conditions to guarantee convergence of the optimization algorithm [70].

However, despite these advancements, there exists a fundamental gap in the application of trust-region and multifidelity model-management frameworks to topology optimization: acceptance metrics currently utilized for both objective-function agreement and error-estimator bounds do not include direct verification that governing PDEs are satisfied. Material-distribution discontinuities and possible equilibrium-violations in coarser mesh proposals require physics-based correctness checks beyond convergence of optimization objectives. Therefore, this research gap motivates the development of the multifidelity acceptance safeguard presented in Chapter 3, which embeds PDE residual-based verification directly into the optimization loop without the need for expensive dual solves or certified error estimators.

2.3 Machine Learning for Topology Optimization

ML is increasingly influencing topology optimization, as evidenced in recent literature. Neural networks have been used either to (i) directly predict optimized topologies, or (ii) accelerate conventional optimization by providing an informed initialization point. Despite this potential, several open issues remain, including: the extent to which learned models generalize across boundary conditions and problem settings, the large training datasets often required to achieve reliable accuracy, and the need for rigorous validation of ML-assisted designs under conditions beyond those represented in the training data [14, 15, 86]. ML roles in topology optimization are demonstrated in Figure 2.4.

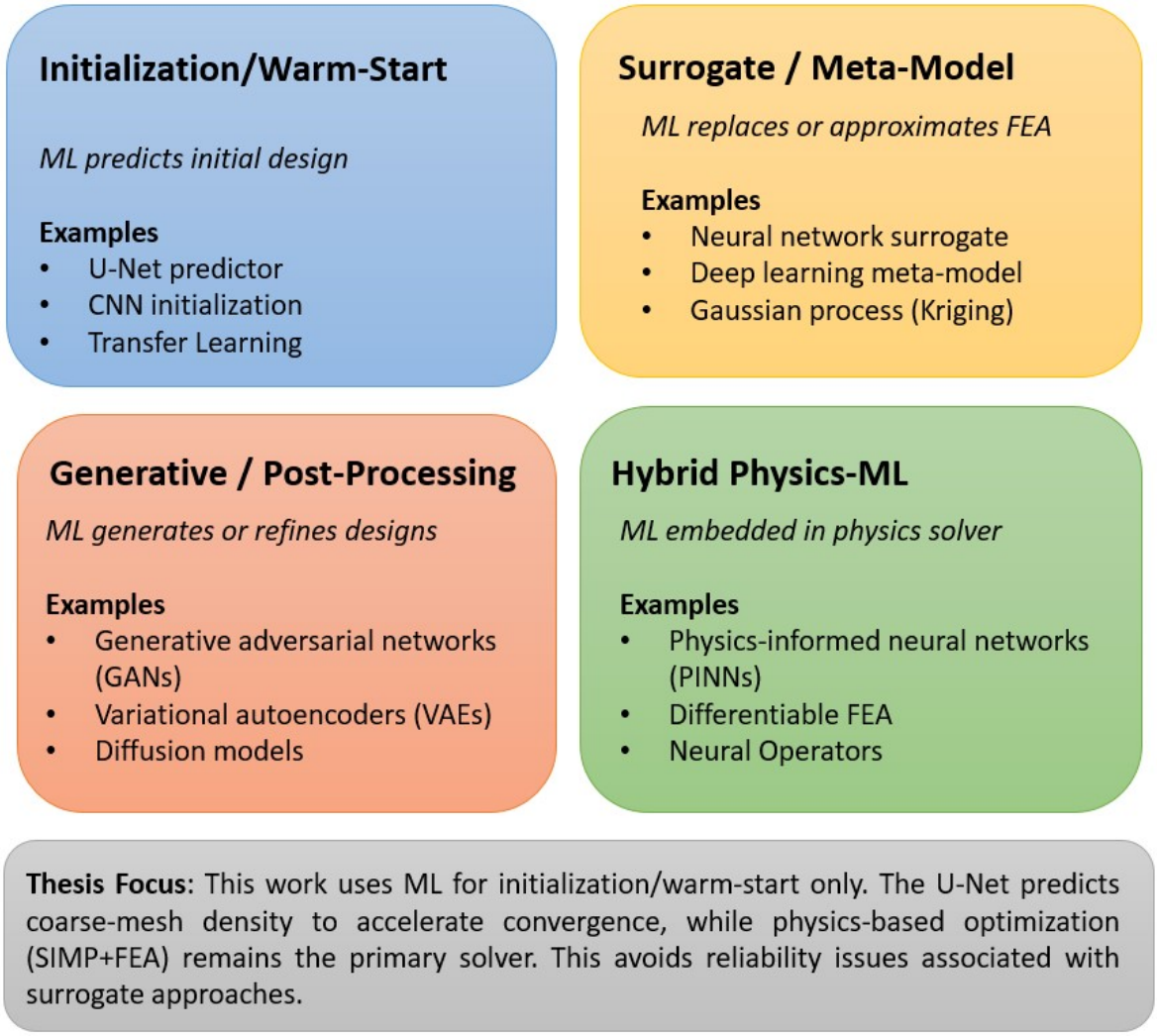

Figure 2.4: ML roles in topology optimization

2.3.1 ML as Initialization/Warm Start

Neural-network-based initialization methods provide an informed starting design while still relying on a physics-based topology optimization procedure as the primary solver. Chandrasekhar et al. [13] introduced TOuNN, which employs a multilayer perceptron to parameterize the density field while continuing to use finite element analysis to enforce equilibrium throughout the optimization. This preserves physics-based verification via repeated finite element solves. The approach was later extended using a Fourier-space projection, enabling length-scale control by mapping spatial coordinates to the Fourier domain prior to neural-network evaluation, thereby reducing reliance on additional explicit length-scale constraints [87].

Chen et al. [88] investigated the use of strain energy fields as conditioning inputs for initialization, reporting faster convergence while maintaining comparable solution quality in their experiments. Zhang et al. [89] proposed a neural reparameterization method (TONR) in which network parameters replace traditional element-wise density

variables; they also employed pre-training to produce an initial near-uniform density distribution consistent with the target volume fraction prior to the full optimization. Kuszczak et al. [90] applied model-agnostic meta-learning to learn neural-network initializations that can adapt to a new design task in fewer iterations, leveraging reusable information from partial optimizations across a range of loading conditions. In these warm-start and reparameterization-based approaches, mechanical feasibility is still assessed through continued finite element analysis during the optimization process, which distinguishes them from surrogate-model approaches aimed at bypassing iterative optimization [13, 88].

2.3.2 ML as Surrogate/Meta-Model

Data-driven surrogate models aim to learn a direct mapping from boundary conditions (and related problem descriptors) to optimized topologies, so that the iterative optimization loop can be reduced or avoided. Yu et al. [12] developed a two-stage approach combining convolutional neural networks (CNNs) with conditional generative adversarial networks (cGANs): a CNN encoder–decoder predicts low-resolution structures from boundary conditions, followed by a cGAN-based refinement stage to produce high-resolution outputs, enabling near-instantaneous prediction after offline training. As noted in the literature, however, such approaches can face significant generalization challenges, with predicted layouts potentially containing disconnected features and exhibiting poor transfer to unseen loading or boundary conditions [14, 15].

In addition to two-stage pipelines, encoder–decoder architectures are widely used in surrogate modeling [15, 86]. For example, Kollmann et al. [91] used ResUNet for metamaterial design, combining residual learning with U-Net skip connections to improve training stability while enabling efficient information propagation. Similarly, Abueidda et al. [92] employed ResUNet for stress-constrained and geometrically nonlinear problems and reported strong pixel-wise density prediction performance. Islam et al. [93] and Rasulzade et al. [94] also found ResUNet to perform well for high-resolution topology prediction, attributing this in part to residual blocks that facilitate training of deeper networks.

Related work explores implicit neural representations as an alternative parameterization of topology fields. Zehnder et al. [95] developed NTopo, which uses periodic activation functions to represent density (and displacement) fields in a mesh-free setting, yielding a neural-representation-based topology optimization formulation.

Transfer learning represents one approach for improving the generalizability of surrogate models [96]. Tan et al. [97] proposed transferring weights from networks trained on coarse-scale data and fine-tuning using only a small number of fine-scale simulations, showing improved generalizability across scales. Propp et al. [98] further

demonstrated transfer learning across multiple surrogate modeling tasks, including cross-dimensional settings.

Despite the growing capabilities of surrogate and meta-modeling approaches, several limitations remain. Many methods require extensive training datasets (often ranging from tens of thousands to over 100,000 samples) to reach acceptable accuracy [14, 99], and generating high-quality training data for large-scale designs can be computationally expensive [93, 100]. The literature also emphasizes challenges in out-of-distribution generalization and the importance of evaluation protocols that test performance beyond the training conditions [15, 86].

2.3.3 Machine Learning for Generative Design/Post-Processing

Machine Learning is being used in addition to providing initialization, for the purpose of supporting generative design workflows and post-processing refinements in topology optimization. Among other tools, Generative Adversarial Networks (GANs) have been shown to be effective for creating multiple design alternatives. Nie et al. [101] created TopologyGAN, a tool based on conditional GANs with physical fields as inputs for direct creation of optimized topologies from boundary conditions. In a similar manner, Kazemi et al. [102] also demonstrated the use of multiphysics GANs for design optimization, where they were able to create concepts for coupled problems using knowledge learned from single-physics solutions. Training instability has been mitigated by the application of the WGAN-GP architecture with gradient penalty mechanisms [103, 104]. Padmaprabhan et al. [105] then developed GO-GAN with new input mechanisms for the encoding of scalar conditions in geometry optimization.

The super-resolution GAN has been used to address the issue of low resolution computations in topology design by increasing the resolution of low resolution predictions. Yu et al. [12] developed a two-stage framework, which combined CNN-based encoders with conditional GANs for progressive refinement. This resulted in high resolution topology designs that were nearly optimal at a minimal computational expense. Zeng et al. [104] improved the structural clarity of the topology designs by applying Pix2pix GAN as a post-processing refinement step after CWGAN-GP generation, thereby improving the prediction accuracy of the generated complex structure.

Generative models based on diffusion modeling represent an emerging alternative to GANs for generative topology design. Zhang et al. [106] applied diffusion models to topology optimization and utilized their training stability advantage over GANs. Giannone et al. [107] investigated diffusion-based approaches to generate optimal topologies via denoising process. Gao et al. [108, 109] combined diffusion models with GANs and attention mechanism to improve training stability and extract structural features from generated topology designs in real time.

Variational Autoencoders (VAEs) have been used to support design exploration and novelty evaluation of generated designs. Oh et al. [110] used VAEs with topology optimization to iteratively design wheels and evaluated the novelty of each design iteration using the reconstruction error of the autoencoder. Regenwetter et al. [111] conducted a comprehensive literature review and found the VAEs provide several benefits including maintaining structured latent spaces, but noted that compared to GANs, VAEs have difficulty in generating high fidelity outputs. It has been emphasized that post-processing the generated designs using physics-based optimization methods remains essential to ensure the structural feasibility and performance of the generated designs [14, 15].

2.3.4 Hybrid Physics – ML

Physics-informed machine learning techniques combine the physical governing equations into neural network structures, providing an alternative to data-only approaches. A physics-informed neural network (PINN) was introduced by Raissi et al. [112] to include physical laws into the loss function of the neural network by converting partial differential equations into constraints allowing for the solution of both forward and inverse problems with very little training data. The use of PINNs offers advantages over typical ML surrogates since they do not require large amounts of labeled data but instead utilize automatic differentiation to maintain the physics of the problem [113].

PINNs are commonly used in topology optimization through the implementation of two types of formulations: PDE-based formulations that minimize residuals of the governing equations, and energy-based formulations that minimize the system's total potential energy [114, 115, 116, 117]. In particular, energy-based PINNs such as the deep energy method show significant computational efficiency improvements over other PINNs due to reduced-order derivatives [118, 119]. Additionally, dynamically configured PINNs, in which the network architecture (and thus the number of trainable parameters) is adapted during the optimization process, have been reported to further reduce computational cost. By allocating model capacity where the solution complexity demands it and avoiding over-parameterization elsewhere, these approaches can improve training efficiency without sacrificing physical consistency. [113].

Another type of hybrid physics–ML technique uses neural operators, which include DeepONet and Fourier Neural Operators, to learn the mapping from one function space to another. This allows the replacement of PDE solvers in iterative optimization frameworks such as SIMP, resulting in a significant reduction in computation time while maintaining the necessary gradient information for sensitivity analysis. In structural topology optimization, this operator-learning perspective is particularly attractive because it can approximate the solution operator from design and loading fields to

displacement (or compliance) responses, enabling fast evaluations within gradient-based design updates [120, 121, 122, 123].

2.4 Verification and Reproducibility in Topology Optimization

2.4.1 Verification Aspects in Topology Optimization

Accelerated approaches in multifidelity topology optimization—where the dominant cost arises from repeated PDE (FEA) solves of the equilibrium equations—utilize lower-cost models to minimize runtimes while still enabling recourse to higher-fidelity models as necessary to ensure accuracy [65]. However, reliability risks arise when lower-fidelity approximations are used repeatedly without explicit verification or model-management safeguards [66]. In multifidelity surrogate modeling, a key challenge is balancing computational efficiency with solution reliability. Moreover, approximation error can propagate across iterations if not controlled through verification or correction steps. Forrester et al. [48] also demonstrated that correlation between the results of low-fidelity and high-fidelity analysis must be managed to avoid model mismatch that can mislead the search and degrade solution quality.

Peherstorfer et al. [65] provided a classification system of multifidelity model management techniques into three categories: adaptation (utilizing sparse high-fidelity data to correct low-fidelity models); fusion (combining information from multiple evaluations); and filtering (screening/selection mechanisms for model/evaluation choice). Trust-region ideas provide systematic accept/reject rules and update the trust-region size based on agreement between predicted and realized improvement [48, 66]. However, in many multifidelity workflows, acceptance is based primarily on objective/response agreement rather than direct verification that the governing equations remain satisfied.

2.4.2 Reproducibility Practices in Topology Optimization

Reproducibility has become a significant concern in computational topology optimization. Mukherjee et al. [5] pointed out that prohibitive computational costs have prevented historical comparative studies across various acceleration methods. Wang et al. [124] stated that the contributions of educators who have made available openly accessible implementations have greatly reduced the barriers to entry and enabled comparative studies. However, Ramu et al. [86] reported that machine learning-based applications in structural optimization often do not use standardized evaluation protocols, with issues being associated with data availability, data quality, and a lack of sufficient validation on out-of-distribution test cases.

2.5 Research Gaps

2.5.1 Lack of Systematic Acceptance Safeguards in Multifidelity Acceleration

Common multifidelity and acceleration strategies manage fidelity transitions using predetermined schedules, error-estimation procedures (often requiring additional solves), or offline-trained surrogates [5]. Although, Peherstorfer et al. [65] emphasized that multifidelity methods require model management strategies that provide theoretical guarantees of accuracy and convergence, systematic runtime verification mechanisms that ensure low-fidelity proposals meet governing physics are relatively rare. Kontogiannis et al. [66] demonstrated that multifidelity surrogate modeling can reduce computational cost, however, their framework—similarly to most of the other methods—does not explicitly emphasize runtime accept/reject safeguards based on direct checks of governing-physics consistency for low-fidelity proposals.

2.5.2 ML Integration Lacks Runtime Verification

There have been numerous applications of machine learning methods for topology optimization. These include surrogate modeling, design initialization, and generative design [14, 86]. However, since machine learning methods are inherently reliant on data, data-driven approaches generate reliability concerns when used without explicit verification against higher-fidelity models. Ramu et al. [86] reported that many of the machine learning approaches were challenged by inaccuracies in prediction, difficulty in generalization, and large amounts of training data required. Mukherjee et al. [5] noted that many ML-based studies are evaluated primarily as stand-alone predictors or accelerators, and that rigorous integration with verification safeguards remains comparatively less developed. In addition, it was reported that generated topologies could be composed of disconnected regions and exhibit poor transferability to unseen conditions. Currently, machine learning integration often functions in an open-loop manner, thereby creating black-box risk where predictions may seem plausible, but fail to satisfy equilibrium conditions at higher fidelity [14, 86].

2.5.3 Limited Cross-Dimensional (2D + 3D) Validation

Cross-dimensional (2D+3D) validation remains underdeveloped. Mukherjee et al. [5] stressed that comparative studies that systematically evaluate acceleration methods across different dimensions of problem space are less common due to the significantly high computational costs involved. Wang et al. [124] indicated that educational contributions are usually focused on specific problem classes and rarely involve systematic

Table 2.2: Research gaps in topology optimization literature

Gap	Current State	Limitation	Thesis Contribution
Systematic Acceptance Safeguards [65, 66]	Fidelity transitions via iteration schedules, error estimates requiring dual solves, or offline training	Accept/reject logic based on governing equation residuals computed from low-fidelity solutions is not widely incorporated explicitly in current multifidelity TO frameworks	MFAS with periodic high-fidelity verification, accept/reject decisions based on compliance, rejection–recovery mechanism
ML Verification [5, 14]	ML predictions trusted implicitly or validated only in post-processing; open-loop integration	Black-box risk: proposals may appear plausible yet fail equilibrium/-constraints at higher fidelity	U-Net initialization subject to same verification criteria as coarse-mesh proposals
2D+3D Validation with Statistics [14, 15, 86]	Methods often validated on single dimensionality or problem class; limited ablation studies	Scarce comparative studies across dimensionality; insufficient statistical rigor (single seeds, limited test cases)	Systematic 2D+3D benchmarking with ablation studies; paired-seed experimental design for statistical assessment

cross-validation. Ramu et al. [14, 86] noted data-related challenges. Similar statistical validation concerns exist for any acceleration strategy evaluated across limited test cases. Ablation studies that quantify sensitivity to verification frequency, acceptance thresholds, or model-hierarchy configuration are still relatively limited in literature [14, 15, 86].

2.5.4 Research Positioning

This research fills the identified gaps by developing a runtime-verified multifidelity framework that includes acceptance safeguards embedded into the topology optimization loop. The research gaps addressed in this thesis are summarized in Table 2.2. The Multifidelity Acceptance Safeguard (MFAS) approach provides for periodic high-fidelity verification, decision-making about whether to accept proposals based on compliance evaluation, and rejection-recovery capabilities to preclude coarse-model error accumulation. The machine learning component remains subject to the same verification criteria as every coarse-mesh proposal, reducing black-box risk and enabling its contribution to be measured empirically rather than assumed. The framework is systematically validated across both two-dimensional and three-dimensional benchmarks, with the inclusion of ablation studies on verification schedules and acceptance thresholds, and employing paired-seed experimental designs for statistical assessment of speed-quality trade-offs.

Chapter 3

Methodology

3.1 Problem Formulation

This chapter presents the methodology for multifidelity acceptance safeguard (MFAS) framework developed in this dissertation. The current section states the optimization problem and defines the two discretization levels (fine and coarse) that every subsequent algorithmic component relies on. Material interpolation, sensitivity analysis, and regularization are discussed in Sections 3.2–3.3, while the MFAS algorithmic loop and its acceptance logic are detailed in Section 3.4.

3.1.1 Compliance Minimization Under a Volume Constraint

The design domain Ω is discretized into N finite elements. Each element e is assigned a density design variable $\rho_e \in [0, 1]$. In the density pipeline adopted in this dissertation, the design variable field is first filtered to obtain the filtered density $\tilde{\rho}_e$, and subsequently projected to obtain the physical (projected) density $\bar{\rho}_e$. Here, "physical" denotes the density field passed to the finite element solver — not a mass density with physical units. Like ρ_e and $\tilde{\rho}_e$, the field $\bar{\rho}_e$ is dimensionless and bounded in $[0, 1]$, representing a local volume fraction of solid material. The operators defining this mapping are introduced in Section 3.3. The physical density $\bar{\rho}_e$ is the quantity used in material interpolation and in the evaluation of the volume constraint, and it determines the stiffness matrix employed in compliance evaluation.

Following the standard power-law formulation [6], the topology optimization problem is stated as

$$
\begin{aligned}
\min_{\boldsymbol{\rho}} \quad & C(\boldsymbol{\rho}) \;=\; \mathbf{u}^{\top}\,\mathbf{K}(\bar{\boldsymbol{\rho}})\,\mathbf{u} \\
\text{subject to} \quad & \frac{V(\bar{\boldsymbol{\rho}})}{V_0} \;\le\; v_f\,, \\
& \mathbf{K}(\bar{\boldsymbol{\rho}})\,\mathbf{u} \;=\; \mathbf{F}\,, \\
& 0 < \rho_{\min} \;\le\; \rho_e \;\le\; 1\,, \qquad e = 1,\dots,N\,,
\end{aligned}
\tag{3.1}
$$

where $\mathbf{u}$ and $\mathbf{F}$ are correspondingly the global displacement and force vectors respectively, $\mathbf{K}(\bar{\boldsymbol{\rho}})$ is the global stiffness matrix assembled from element contributions that depend on the physical densities $\bar{\boldsymbol{\rho}}$, $V(\bar{\boldsymbol{\rho}})$ is the total material volume, V_0 is the design-domain volume, and v_f is the prescribed volume fraction. Compliance C is minimized on the raw design variable $\boldsymbol{\rho}$; its dependence on $\bar{\boldsymbol{\rho}}$ is implicit through the density pipeline $\boldsymbol{\rho} \to \tilde{\boldsymbol{\rho}} \to \bar{\boldsymbol{\rho}}$ defined in Section 3.3. The lower bound $\rho_{\min}$ is a small positive constant that prevents numerical singularities in the OC update and avoids exact void densities. The SIMP interpolation introduces a void-stiffness regularization parameter $E_{\min}$ to prevent singularity of $\mathbf{K}$.

For a discretization with element volumes v_e, the volume constraint reads [6]

$$
V(\bar{\boldsymbol{\rho}}) \;=\; \sum_{e=1}^{N} v_e\,\bar{\rho}_e \;\le\; v_f\,V_0\,. \tag{3.2}
$$

Here $\bar{\rho}_e \in [0,1]$ is a dimensionless density variable representing the local solid-volume fraction in element e, not a physical mass density. Consequently, $v_e\bar{\rho}_e$ has units of volume, and $V(\bar{\boldsymbol{\rho}})$ has the same units as V_0, so Eq. 3.2 is dimensionally consistent. Equivalently, the normalized constraint $V(\bar{\boldsymbol{\rho}})/V_0 \le v_f$ is dimensionless on both sides. For uniform elements (square in 2D, cubic in 3D), $v_e = V_0/N$ for all e. For a fully designable domain, the constraint reduces to a bound on the arithmetic mean of $\bar{\rho}_e$. When passive (non-designable) elements are present—as in the three-dimensional L-bracket benchmark—the summation in Eq. (3.2) is restricted to the active element set, with passive void elements excluded via an element mask. Although the constraint is stated as $V(\bar{\boldsymbol{\rho}})/V_0 \le v_f$, the OC update typically drives it to an active constraint, yielding $V(\bar{\boldsymbol{\rho}})/V_0 = v_f$ at convergence. Here $N = n_{\text{elx}} \times n_{\text{ely}}$ (2D) and $N = n_{\text{elx}} \times n_{\text{ely}} \times n_{\text{elz}}$ (3D) is the total number of finite elements used to discretize the design domain.

Unless otherwise specified, the numerical studies in this dissertation employ the common nondimensionalization used in SIMP benchmark settings: element size is taken as the unit length, the solid stiffness is normalized (typically $E_0 = 1$), and loads are scaled consistently so that compliance values are interpreted as relative (dimensionless) measures for method comparison [6, 51].

3.1.2 Fine and Coarse Discretizations

The multifidelity framework operates on two nested discretizations of the same rectangular design domain. A *strict* $2\times$ *refinement rule* is adopted: each spatial dimension of the coarse mesh is exactly half the resolution of the fine mesh. For two-dimensional problems, the fine mesh consists of $n_x^f \times n_y^f$ elements with a total of $N_f = n_x^f \times n_y^f$ elements. The coarse mesh has $n_x^c = n_x^f/2$ and $n_y^c = n_y^f/2$ elements, so that $N_c = N_f/4$. Each coarse element corresponds to exactly $2 \times 2 = 4$ fine elements. For three-dimensional problems, the fine mesh consists of $n_x^f \times n_y^f \times n_z^f$ elements. The coarse mesh has half the resolution in each coordinate direction, yielding $N_c = N_f/8$. Each coarse element maps to a $2\times2\times2$ block of fine elements. Throughout, superscripts f and c on element counts (e.g. n_x^f, n_y^c) denote fine and coarse meshes respectively; subscripts x, y, z identify the coordinate direction.

The strict $2\times$ assumption ensures that the restriction operator R (fine $\rightarrow$ coarse) and the prolongation operator P (coarse $\rightarrow$ fine) are structurally simple and parameter-free. Here R denotes the restriction operator that transfers element-wise fields from the fine mesh to the coarse mesh, and P denotes the prolongation operator that transfers element-wise fields from the coarse mesh to the fine mesh. Because the design domain geometry and element aspect ratios are preserved across levels, boundary conditions and applied loads can be imposed consistently on both meshes without geometric interpolation.

For a given density field, a *fine-mesh compliance evaluation* assembles the stiffness matrix $\mathbf{K}(\bar{\boldsymbol{\rho}}_f)$ on the fine mesh, solves the equilibrium system $\mathbf{K}(\bar{\boldsymbol{\rho}}_f)\,\mathbf{u}_f = \mathbf{F}_f$, and computes compliance using the following mathematical formula:

$$C_f \;=\; \mathbf{u}_f^\top \mathbf{K}(\bar{\boldsymbol{\rho}}_f)\,\mathbf{u}_f\,. \tag{3.3}$$

Here the subscript f denotes fine-mesh quantities; $\mathbf{u}_f$ and $\mathbf{F}_f$ are the fine-level global displacement and force vectors, respectively. A *coarse-mesh compliance evaluation* proceeds analogously on the coarse mesh. The fine-mesh compliance C_f is the authoritative measure of structural performance used in all reported results and in the MFAS verification/acceptance logic. Coarse-mesh compliance is used only as an inexpensive proxy between verification points.

When a density field originates on one mesh level and must be evaluated on the other, it is first transferred via P or R and then processed through the density pipeline (filtering and, where applicable, projection) on the target mesh before compliance is computed. Volume enforcement is applied after inter-mesh transfer to ensure that the volume constraint remains satisfied on the evaluation mesh.

3.2 Finite Element Model and SIMP Interpolation

This section specifies the material interpolation law, and the compliance sensitivity expression as implemented in the thesis solver. These components are shared by every method variant (M1–M4) defined in Section 3.6 and are evaluated on both the fine and coarse meshes introduced in Section 3.1. Throughout this dissertation, the term *PDE-constrained topology optimization* refers to the fact that for the equilibrium constraint

$$\mathbf{K}(\bar{\boldsymbol{\rho}})\,\mathbf{u} = \mathbf{F}\,, \tag{3.4}$$

the finite element discretization of the linear elasticity boundary-value problem on Ω — must be satisfied exactly at every design iteration, not merely at convergence.

3.2.1 SIMP Material Interpolation

As introduced in Section 2.1.2, the SIMP (Solid Isotropic Material with Penalization) method interpolates element stiffness via a penalized power law to discourage intermediate densities and promote binary topologies. This section specifies the formulation adopted in this thesis.

The stiffness of each element is interpolated using the modified SIMP power law [4, 16, 23],

$$E_e(\bar{\rho}_e) \;=\; E_{\min} \;+\; \bar{\rho}_e^{\,p}\left(E_0 - E_{\min}\right), \tag{3.5}$$

where E_0 is the Young's modulus of the solid material (units of stress, e.g. MPa), $E_{\min} \ll E_0$ is a small ersatz modulus (units of Pa, same as E_0) added to all elements to prevent singularity of $\mathbf{K}$, and p is the penalization exponent. The subscript e on E_e denotes the element index; the function itself depends only on the scalar density $\bar{\rho}_e$ and is identical for all elements of the same density value. In the non-dimensionalized benchmark setting adopted in this thesis, $E_0 = 1$ and stiffness values are expressed relative to this reference, so compliance $C = \mathbf{u}^\top\mathbf{K}\mathbf{u}$ is dimensionless. In the present work, Eq. (3.5) is evaluated using the *physical density* $\bar{\rho}_e$ produced by the density filter and projection operators defined in Section 3.3, not the raw design variable ρ_e. This ensures that the material interpolation respects the regularization imposed by the density pipeline.

The element stiffness matrix is written as

$$\mathbf{K}_e(\bar{\rho}_e) \;=\; E_e(\bar{\rho}_e)\,\mathbf{k}_0\,, \tag{3.6}$$

where $\mathbf{k}_0$ is the unit-modulus element stiffness matrix (for the chosen element type and Poisson's ratio). The global stiffness matrix is assembled by summing element

contributions:

$$\mathbf{K}(\bar{\boldsymbol{\rho}}) \;=\; \sum_{e=1}^{N} \mathbf{K}_e(\bar{\rho}_e)\,. \tag{3.7}$$

The parameter values used in all numerical experiments are $E_0 = 1.0$ (normalized) and $p = 3.0$ unless otherwise stated. In the two-dimensional solver $E_{\min} = 10^{-3}$; in the three-dimensional solver $E_{\min} = 10^{-6}$. Both values are small enough to prevent singularity of $\mathbf{K}$ while having negligible effect on the optimized topology. These choices follow the standard benchmark settings established in the literature [6, 16, 51].

3.2.2 Compliance and Sensitivity

Substituting the element-level interpolation into the compliance objective stated in Eq. (3.1) gives the element-sum representation. The compliance can be written as [6, 16]

$$C(\bar{\boldsymbol{\rho}}) \;=\; \mathbf{u}^{\top}\,\mathbf{K}(\bar{\boldsymbol{\rho}})\,\mathbf{u} \;=\; \sum_{e=1}^{N} E_e(\bar{\rho}_e)\;\mathbf{u}_e^{\top}\,\mathbf{k}_0\,\mathbf{u}_e\,, \tag{3.8}$$

where $\mathbf{u}_e$ denotes the element displacement vector extracted from $\mathbf{u}$. Because the problem is self-adjoint, the sensitivity of the compliance with respect to the physical density of element e follows by direct differentiation. Adapting the classical result of Sigmund [6] to the interpolation of Eq. (3.5) yields

$$\frac{\partial C}{\partial \bar{\rho}_e} \;=\; -\,p\;\bar{\rho}_e^{\,p-1}\,\bigl(E_0 - E_{\min}\bigr)\;\mathbf{u}_e^{\top}\,\mathbf{k}_0\,\mathbf{u}_e\,. \tag{3.9}$$

The negative sign confirms that increasing the physical density of any element reduces the compliance (i.e., stiffens the structure). Equation (3.9) is evaluated at every iteration for every active element on whichever mesh (fine or coarse) the current method variant operates. Because the density pipeline introduces filtering and projection upstream of $\bar{\rho}_e$, the chain rule through those operators is required for the full design-variable sensitivity; this is addressed in Section 3.3.

3.2.3 Element Types and Linear Solver

In two-dimensional problems, the design domain is discretized with four-node bilinear quadrilateral (Q4) elements under plane stress [125, 126]. Each node carries two translational degrees of freedom, yielding an 8×8 element stiffness matrix $\mathbf{k}_0$. The equilibrium system (3.4) is solved by a sparse direct solver.

In three-dimensional problems, the design domain is discretized with eight-node trilinear hexahedral (H8) elements [125, 126]. Each node carries three translational degrees of freedom, yielding a 24×24 element stiffness matrix $\mathbf{k}_0$. Because the resulting global system is substantially larger than in the two-dimensional case, Eq. (3.4) is solved

using a preconditioned conjugate gradient (PCG) method with a Jacobi preconditioner [5, 51]. A convergence tolerance of 10^{-6} on the relative residual norm is enforced. If the iterative solver fails to converge within a prescribed maximum iteration count, the PCG solve is restarted with a relaxed tolerance and an increased iteration limit.

3.3 Regularization and Optimization Update Scheme

This section defines the density filter and projection operators that regularize the optimization problem, states the canonical density pipeline, and presents the Optimality Criteria (OC) update rule together with the move-limit schedule and convergence criterion employed in the implementation.

3.3.1 Density Filter

Mesh independence and suppression of checkerboard artifacts are ensured by a density filter applied to the design variables before they enter the material interpolation. For every element e the filtered density $\tilde{\rho}_e$ is obtained as the normalized, distance-weighted average of the design variables in a neighborhood $\mathcal{N}_e$. For uniform element volumes ($v_e = \text{const.}$), the element volumes cancel and the density filter reduces to (with uniform v_j) [20, 43]

$$\tilde{\rho}_e \;=\; \frac{\displaystyle\sum_{i \in \mathcal{N}_e} H_{ei}\, \rho_i}{\displaystyle\sum_{i \in \mathcal{N}_e} H_{ei}}\,, \tag{3.10}$$

where $\mathcal{N}_e$ is the set of elements whose center-to-center distance $\operatorname{dist}(e,\, i)$ to element e does not exceed the filter radius $r_{\min}$:

$$\mathcal{N}_e \;=\; \big\{\, i \;:\; \operatorname{dist}(e,\, i) \;\le\; r_{\min} \big\}\,, \tag{3.11}$$

and the weight factor is defined as a function of the distance between neighboring elements:

$$H_{ei} \;=\; \max\!\big(0,\; r_{\min} - \operatorname{dist}(e,\, i)\big)\,. \tag{3.12}$$

The combination of Eqs. (3.10)–(3.12) define the filter operator $\mathcal{F}_{r_{\min}}$, so that $\tilde{\boldsymbol{\rho}} = \mathcal{F}_{r_{\min}}(\boldsymbol{\rho})$. In matrix notation the element-wise filter is realized as [16]

$$\tilde{\boldsymbol{\rho}} \;=\; \mathbf{H}\,\boldsymbol{\rho} \;\oslash\; \mathbf{H}_s\,, \tag{3.13}$$

where $\mathbf{H} \in \mathbb{R}^{N \times N}$ is the sparse matrix with entries H_{ei}, $\mathbf{H}_s \in \mathbb{R}^N$ is the row-sum vector with components $(\mathbf{H}_s)_e = \sum_{i \in \mathcal{N}_e} H_{ei}$, and $\oslash$ denotes element-wise (Hadamard) division. Because the multifidelity framework operates on two mesh levels (Section 3.1.2), the

filter is instantiated once per level with its own radius. On the fine mesh the radius is $r_{\min,f}$; on the coarse mesh it is set to $r_{\min,c} = \max(1,\ r_{\min,f}/2)$, which preserves the absolute physical extent of the filter support across levels (each mesh level has the same product $r_{\min} \times h_e$, where h_e is the element edge length on that level).

In the two-dimensional solver the filter is assembled as a sparse matrix $\mathbf{H}$ with the row-sum normalization vector $\mathbf{H}_s$ precomputed once before the optimization loop. In the three-dimensional solver the same cone-shaped kernel is stored as a small dense tensor and the filtering is realized as a three-dimensional convolution, followed by element-wise division by the analogous row-sum field to correct for boundary truncation. Both implementations produce results that are algebraically equivalent to Eq. (3.10).

In the general density pipeline the filtered density $\tilde{\rho}_e$ may be further processed by a smoothed Heaviside step function to promote near-binary designs [80, 127]. The projected (physical) density is defined as:

$$\bar{\rho}_e \;=\; 1 - e^{-\beta\,\tilde{\rho}_e} \;+\; \tilde{\rho}_e\, e^{-\beta}\,, \tag{3.14}$$

where β is the sharpness parameter controlling the degree of projection. When $\beta = 0$ the projection reduces to the identity ($\bar{\rho}_e = \tilde{\rho}_e$), recovering the pure density filter; as $\beta \to \infty$ the projection approaches a true Heaviside step at $\tilde{\rho}_e = 0$.

A continuation scheme is conventionally used in which β is gradually increased—for example, doubled every 50 iterations or whenever the design-variable change drops below a prescribed threshold. The derivative required for the sensitivity chain rule is [16]

$$\frac{\partial \bar{\rho}_e}{\partial \tilde{\rho}_e} \;=\; \beta\, e^{-\beta\,\tilde{\rho}_e} \;+\; e^{-\beta}\,. \tag{3.15}$$

In the present thesis, both the two-dimensional and three-dimensional implementations employ the density filter alone with no active Heaviside projection; that is, $\beta = 0$ throughout, so $\bar{\rho}_e = \tilde{\rho}_e$ and the projection operator $\mathcal{H}_\beta$ acts as the identity. The notation $\bar{\rho}_e$ is nevertheless retained to keep the formulation general and to facilitate future extensions that activate projection continuation.

3.3.2 Density Pipeline

The complete density pipeline is obtained by combining the filter and projection operators. The design variable ρ_e is filtered to the intermediate density $\tilde{\rho}_e = \mathcal{F}_{r_{\min}}(\rho_e)$ and then projected to the physical density $\bar{\rho}_e = \mathcal{H}_\beta(\tilde{\rho}_e)$. The physical density $\bar{\rho}_e$ is the field that enters the SIMP interpolation (Eq. 3.5), the compliance evaluation (Eq. 3.8), and the volume constraint (Eq. 3.2). As discussed previously, in the current implementation, the projection step reduces to the identity. Density pipeline is illustrated in Figure 3.1

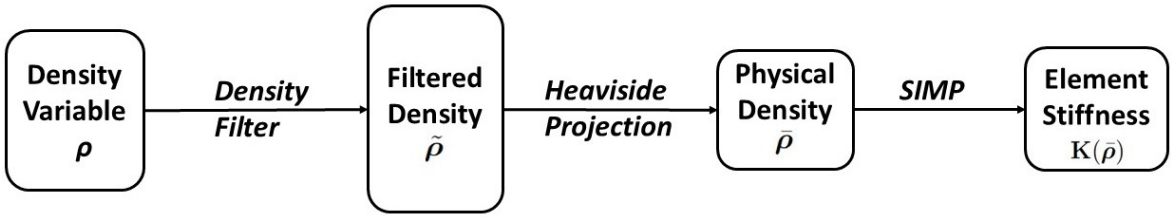

Figure 3.1: Density pipeline

Because the compliance sensitivity in Eq. (3.9) is expressed with respect to the physical density $\bar{\rho}_e$, the chain rule through the density pipeline must be applied to obtain the sensitivity with respect to the design variable ρ_j [16]. For the density filter with no active projection ($\beta = 0$), this chain rule reduces to:

$$\frac{\partial C}{\partial \rho_j} = \sum_{e \in \mathcal{N}_j} \frac{H_{je}}{\sum_{i \in \mathcal{N}_e} H_{ei}} \frac{\partial C}{\partial \tilde{\rho}_e}, \tag{3.16}$$

which, under the current setting $\beta = 0$, uses $\partial C/\partial \tilde{\rho}_e = \partial C/\partial \bar{\rho}_e$ as given in Eq. (3.9). Here $\mathcal{N}_j = \{e : j \in \mathcal{N}_e\}$ denotes the set of elements whose filter neighbourhood includes design variable j, i.e. the transpose of the neighborhood operator $\mathcal{N}_e$. The index pair (j, e) therefore spans the same support as the filter matrix $\mathbf{H}$.

In the two-dimensional solver the chain rule is realized by applying the density filter to the raw sensitivity field—an operation equivalent to Eq. (3.16) in the domain interior and differing only at boundary elements where the row-sum normalization is non-uniform [6, 16, 127]. In the three-dimensional solver the sensitivity modification follows the heuristic filtering concept [51, 127]. ... is convolved with the filter kernel and subsequently divided by the convolution of $\bar{\rho}_e$ alone, yielding the filtered sensitivity $(\widehat{\partial C/\partial \rho_e})$ (hat denotes the heuristically filtered quantity):

$$\frac{\widehat{\partial C}}{\partial \rho_e} = \frac{\sum_{i \in \mathcal{N}_e} H_{ei}\, \bar{\rho}_i \dfrac{\partial C}{\partial \bar{\rho}_i}}{\sum_{i \in \mathcal{N}_e} H_{ei}\, \bar{\rho}_i}. \tag{3.17}$$

This is a density-weighted variant of the classical sensitivity filter; the denominator uses the density-weighted convolution $\sum H_{ei}\bar{\rho}_i$. In the domain interior both forms yield equivalent results and have been extensively validated.

3.3.3 Optimality Criteria Update

The volume-constrained compliance minimization is solved by the standard heuristic Optimality Criteria (OC) method [6, 16, 51]. The design variables are updated element-

wise as

$$\rho_e^{\text{new}} \;=\; \begin{cases} \max\big(\rho_{\min},\ \rho_e - m\big), & \text{if } \ \rho_e\, B_e^{1/2} \;\le\; \max(\rho_{\min},\ \rho_e - m)\,, \\ \min\big(1,\ \rho_e + m\big), & \text{if } \ \rho_e\, B_e^{1/2} \;\ge\; \min(1,\ \rho_e + m)\,, \\ \rho_e\, B_e^{1/2}, & \text{otherwise}\,, \end{cases} \tag{3.18}$$

where m is a positive move limit, the exponent $1/2$ is the numerical damping coefficient, and B_e is the element-level optimality indicator:

$$B_e \;=\; \frac{-\dfrac{\partial C}{\partial \rho_e}}{\lambda \dfrac{\partial V}{\partial \rho_e}}\,, \tag{3.19}$$

in which λ is the Lagrange multiplier for the volume constraint, determined at each iteration by a bisection algorithm so that $V(\bar{\boldsymbol{\rho}})/V_0 = v_f$ is satisfied, and $\partial V/\partial \rho_e$ is the sensitivity of the volume with respect to the raw design variable, obtained by applying the filter chain rule to $\partial V/\partial \bar{\rho}_e = v_e$:

$$\frac{\partial V}{\partial \rho_e} \;=\; v_e\, \frac{\partial \bar{\rho}_e}{\partial \rho_e} \;=\; v_e\, \frac{H_{ee}}{(\mathbf{H}_s)_e}\,, \tag{3.20}$$

where the last equality uses the $\beta = 0$ identity $\partial \bar{\rho}_e/\partial \rho_e = H_{ee}/(\mathbf{H}_s)_e$ from the density filter (Eq. (3.10)). For uniform elements ($v_e = V_0/N$) this is a positive constant, confirming that the bisection for λ is well-posed. The update in Eq. (3.18) with the $1/2$ exponent corresponds to the classical OC scheme that has become standard practice in density-based topology optimization [16, 51]. In both the 2D as well as 3D implementations, the sensitivities entering B_e are the chain-rule-filtered (2D, Eq. (3.16)) or heuristic-filtered (3D, Eq. (3.17)) design-variable sensitivities, not the raw physical-density sensitivities of Eq. (3.9).

In the two-dimensional implementation a fixed move limit $m = 0.2$ is used for all iterations. In the three-dimensional implementation a piecewise-constant schedule reduces the move limit as the optimization progresses, which improves convergence stability at late iterations when changes should be small:

$$m \;=\; \begin{cases} 0.20, & \text{for } k < 150\,, \\ 0.10, & \text{for } 150 \le k < 200\,, \\ 0.05, & \text{for } 200 \le k < 250\,, \\ 0.02, & \text{for } k \ge 250\,, \end{cases} \tag{3.21}$$

where k denotes the iteration counter.

3.3.4 Convergence Criterion

The optimization loop terminates when the maximum element-wise change in the design variable field remains below the tolerance $\mathrm{tol}_{\mathrm{change}}$ for k_{consec} consecutive iterations, subject to a minimum iteration count:

$$\max_e \left|\rho_e^{(k)} - \rho_e^{(k-1)}\right| \;<\; \mathrm{tol}_{\mathrm{change}} \qquad \text{for } k_{\mathrm{consec}} \text{ consecutive iterations}, \quad k \geq k_{\min}\,. \tag{3.22}$$

In both implementations, the density change is evaluated on the filtered (physical) density field $\bar{\rho}_e$, not on the raw design variable ρ_e. Since no heaviside projection is active ($\beta = 0$), these fields are related by the linear density filter, and the distinction is minor in the domain interior. The specific parameter values ($\mathrm{tol}_{\mathrm{change}}$, k_{consec}, $k_{\min}$, and the maximum iteration count) are listed in Chapter 4. If the maximum iteration count is reached before the convergence condition is met, the optimizer is terminated and the current design is accepted as the final result.

3.4 Multifidelity Acceptance Safeguard

The current section defines the multifidelity acceptance safeguard (MFAS) loop that constitutes the core contribution of this thesis. Section 3.4.1 defines the restriction and prolongation operators that transfer density fields between the coarse and fine meshes introduced in Section 3.1.2. Section 3.4.2 presents the MFAS iteration loop and its verification schedule. Section 3.4.3 specifies the acceptance criteria used in the two-dimensional and three-dimensional implementations, respectively.

3.4.1 Mapping Operators Under Strict $2\times$ Refinement

Transfer of density fields between the fine and coarse meshes relies on two complementary mapping operators: a restriction operator R that projects the fine-mesh field onto the coarse mesh, and a prolongation operator P that interpolates the coarse-mesh field back to the fine mesh. Both operators exploit the strict $2\times$ refinement assumption stated in Section 3.1.2, whereby every coarse element corresponds to an integer block of fine elements (a 2×2 block in two dimensions, $2 \times 2 \times 2$ in three dimensions).

The restriction operator maps the fine-mesh density field to the coarse mesh by local averaging. In two dimensions, each coarse element density is obtained as the arithmetic mean of the four fine elements that compose it:

$$\rho^c_{i,j} \;=\; \frac{1}{4}\left(\rho^f_{2i,\,2j} + \rho^f_{2i+1,\,2j} + \rho^f_{2i,\,2j+1} + \rho^f_{2i+1,\,2j+1}\right), \tag{3.23}$$

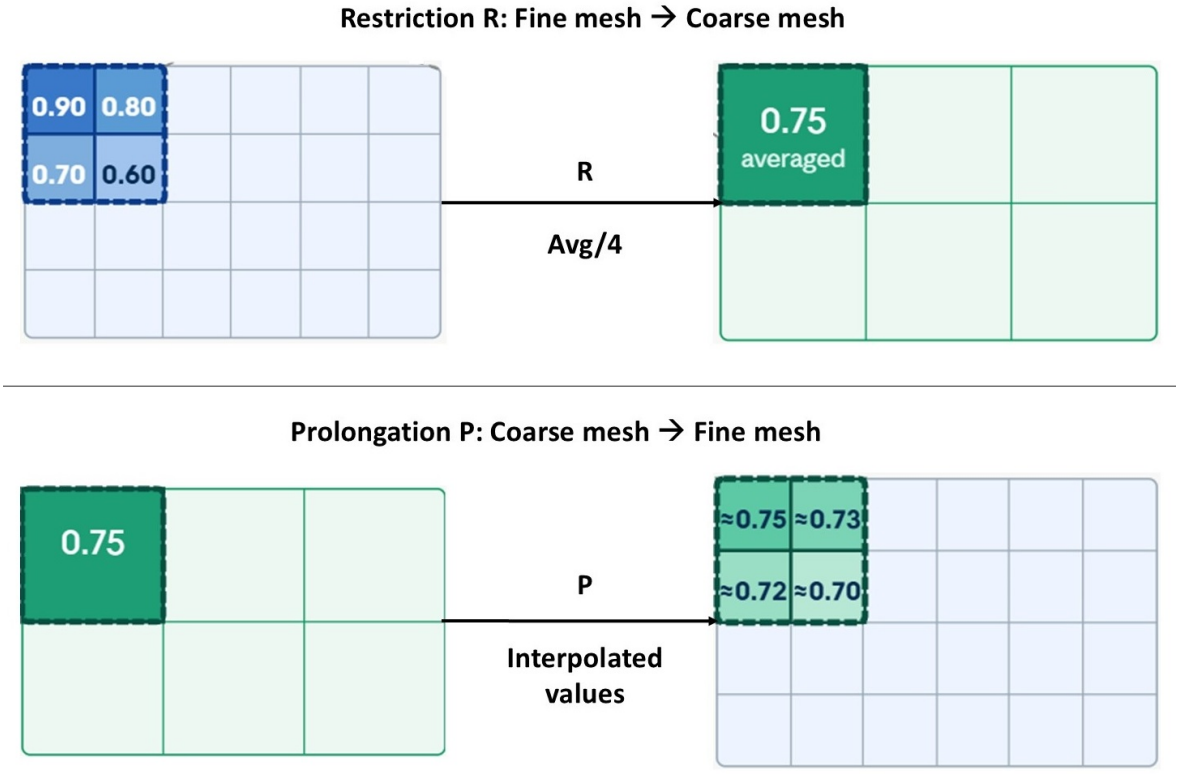

Figure 3.2: Restriction operator R (top) and prolongation operator P (bottom) under the strict 2× refinement rule.

where superscripts f and c distinguish fine- and coarse-mesh fields, and (i, j) denotes the row–column index of the coarse element.

In three dimensions the same averaging principle extends to a $2 \times 2 \times 2$ block of fine elements, implemented as a three-dimensional average-pooling operation with kernel size 2 and stride 2:

$$\rho^{c}_{i,j,k} \;=\; \frac{1}{8} \sum_{\alpha=0}^{1} \sum_{\beta=0}^{1} \sum_{\gamma=0}^{1} \rho^{f}_{2i+\alpha,\, 2j+\beta,\, 2k+\gamma} \,. \tag{3.24}$$

Here $\alpha, \beta, \gamma \in \{0, 1\}$ are local offset indices spanning the $2 \times 2 \times 2$ fine-element block associated with coarse element (i, j, k). Both forms are simple block-averaging operators that are standard in multigrid restriction [128]; no weighting or filtering is applied during restriction. Mapping operators are illustrated in Figure 3.2.

The prolongation operator maps a coarse-mesh density field to the fine mesh by interpolation. In two dimensions, bilinear interpolation is used. The coarse-mesh array is interpolated to twice its resolution using first-order (bilinear) interpolation, producing a fine-mesh field whose resolution matches the fine discretization. In three dimensions, trilinear interpolation is applied: the coarse-mesh array is resized to the fine-mesh dimensions using three-dimensional linear interpolation. Both operators yield continuous density fields on the fine mesh. Because interpolation does not generally preserve the global volume fraction, a volume enforcement step is applied immediately

after prolongation.

After each application of R or P, the mapped density field may violate the prescribed volume fraction v_f. A post-mapping volume enforcement step restores the constraint by uniform multiplicative scaling. Given a mapped field $\hat{\boldsymbol{\rho}}$, the enforced field is

$$\tilde{\boldsymbol{\rho}}_{\text{enf}} \;=\; \text{clamp}\big(s^*\,\hat{\boldsymbol{\rho}},\; \rho_{\min},\; 1\big), \tag{3.25}$$

where s^* is the scalar determined by bisection such that the mean of the clamped field satisfies

$$\frac{1}{N_{\text{act}}}\sum_{e=1}^{N_{\text{act}}} \tilde{\rho}_{\text{enf},e} \;=\; v_f\,. \tag{3.26}$$

For problems that contain passive (void-fixed) regions, such as an L-bracket geometry with a re-entrant corner cutout, elements inside the passive cutout are forced to $\rho_{\min}$ after every mapping and after every OC update [51]. The volume constraint in Eq. (3.2) then effectively applies over the active element set only (with $N_{\text{act}} < N$), ensuring that the void region does not contribute to the material budget.

3.4.2 MFAS Iteration Loop and Verification Schedule

The MFAS framework alternates between inexpensive coarse-mesh optimization iterations and periodic fine-mesh verification checks that decide whether to accept or reject the coarse-mesh proposal. This subsection defines the state variables, the verification schedule, and the overall loop logic.

The safeguard logic maintains the following state throughout the MFAS loop:

- C_{best} — the lowest verified fine-mesh compliance observed so far, and its associated fine-mesh density field $\boldsymbol{\rho}^f_{\text{best}}$;
- C_{prev} — the fine-mesh compliance at the most recent accepted verification (used only in the two-dimensional trust-ratio criterion);
- C_{trial} — the fine-mesh compliance of the current trial design obtained by prolongating the latest coarse update;
- $C_{\text{coarse,old}}$ — the coarse-mesh compliance recorded at the previous verification instant (used only in the two-dimensional trust-ratio criterion);
- $C_{\text{coarse,trial}}$ — the coarse-mesh compliance at the current verification instant.

In the three-dimensional implementation only C_{best} (with its associated $\boldsymbol{\rho}^f_{\text{best}}$ and displacement field $\mathbf{u}_{\text{best}}$) and C_{trial} are required, because the acceptance criterion (Section 3.4.3) does not involve a trust ratio.

The integer parameter f_{verify} controls how often a fine-mesh evaluation is performed. Specifically, after every block of f_{verify} coarse iterations the current coarse design is prolongated to the fine mesh, volume-enforced, and evaluated by a fine-mesh FEA solve, producing C_{trial}. In the two-dimensional implementation the verification occurs at iterations k satisfying $k \bmod f_{\text{verify}} = 0$. In the three-dimensional implementation the verification occurs at the end of each f_{verify}-iteration block, i.e. at iterations k satisfying $(k+1) \bmod f_{\text{verify}} = 0$.

If the trial design is rejected (see Section 3.4.3), the coarse-mesh state is resynchronized to the last accepted fine-mesh state. In the two-dimensional implementation, before resynchronizing, a backtracking procedure (Eq. 3.29) attempts to recover a less conservative step. If backtracking succeeds the candidate is accepted and no resynchronization occurs. Only if all backtracking attempts fail is the resynchronization $\boldsymbol{\rho}^c \leftarrow R(\boldsymbol{\rho}^f_{\text{best}})$ applied, followed by volume enforcement. In the three-dimensional implementation, the fine-mesh state is restored to $\boldsymbol{\rho}^f_{\text{best}}$ (and $\mathbf{u}_{\text{best}}$) on rejection. The coarse-mesh optimizer continues from its current state without restriction. The optimizer thus always resumes from a physics-validated fine-mesh design.

In the two-dimensional criterion (Section 3.4.3), the trust ratio κ requires a nonzero denominator. When the magnitude of the predicted coarse reduction is below a relative numerical tolerance,

$$\left|C_{\text{coarse,old}} - C_{\text{coarse,trial}}\right| \;\le\; 10^{-12}\,\max\!\big(1,\;|C_{\text{prev}}|\big)\,, \tag{3.27}$$

the trust ratio is undefined and not computed. In this case the acceptance decision reduces to a single condition: the trial is accepted if and only if

$$C_{\text{trial}} \;\le\; C_{\text{prev}}\,. \tag{3.28}$$

The sigma-margin condition and the trust-ratio threshold are not applied because the coarse model has provided no useful directional information.

When the initial trial is rejected and $b_{\max} > 0$, the two-dimensional implementation attempts a backtracking recovery before committing to resynchronization. A convex combination of the previous coarse state $\boldsymbol{\rho}^c_{\text{old}}$ and the rejected trial $\boldsymbol{\rho}^c_{\text{new}}$ is formed as follows:

$$\boldsymbol{\rho}^c_{\text{bt}} \;=\; (1-\alpha)\,\boldsymbol{\rho}^c_{\text{old}} \;+\; \alpha\,\boldsymbol{\rho}^c_{\text{new}}\,, \tag{3.29}$$

where α is initialized to α_{shrink} and halved at each attempt ($\alpha \leftarrow \alpha_{\text{shrink}},\ \alpha^2_{\text{shrink}}, \ldots$). Up to $b_{\max}$ candidates are evaluated: each is prolongated to the fine mesh, volume-enforced, and tested against the acceptance criterion (Section 3.4.3). If any candidate passes, it is accepted in place of the original trial. If all $b_{\max}$ attempts fail, resynchronization from $\boldsymbol{\rho}^f_{\text{best}}$ proceeds as described above.

The three-dimensional MFAS loop does not apply the density-change convergence criterion (Eq. 3.22). Instead, it employs a dedicated early-stopping rule: the loop terminates if no improvement in C_{best} has been recorded for n_{stag} consecutive verification cycles, provided that at least $n_{\text{ver,min}}$ total verification evaluations have been performed (default values: $n_{\text{stag}} = 4$, $n_{\text{ver,min}} = 10$). This rule prevents unnecessary coarse iterations after the design has effectively converged at the fine-mesh level.

3.4.3 Acceptance Criteria

The two-dimensional and three-dimensional implementations employ distinct acceptance criteria, reflecting different trade-offs between strictness and computational simplicity. Both criteria are evaluated at every verification step.

The two-dimensional acceptance criterion adapts the classical trust-region ratio introduced by Yuan [129] and applied in multifidelity optimization [48, 65]. The classical trust-region literature defines the agreement ratio as the quotient of actual objective reduction to predicted reduction.

$$r \;=\; \frac{f(\mathbf{x}_{m-1}) \;-\; f(\mathbf{x}_m)}{f(\mathbf{x}_{m-1}) \;-\; \hat{y}(\mathbf{x}_m)}\,, \tag{3.30}$$

which compares the actual reduction in the high-fidelity objective f to the reduction predicted by the surrogate $\hat{y}$. Here, $f(\cdot)$ denotes the high-fidelity objective and $\hat{y}(\cdot)$ its surrogate (low-fidelity) prediction, so that r measures the ratio of actual to predicted decrease. In the present work, this framework is adapted to the multifidelity topology optimization setting by identifying the high-fidelity objective f with the fine-mesh compliance C and the surrogate $\hat{y}$ with the coarse-mesh compliance. The resulting trust ratio κ is:

$$\kappa \;=\; \frac{C_{\text{prev}} - C_{\text{trial}}}{C_{\text{coarse,old}} - C_{\text{coarse,trial}}}\,, \tag{3.31}$$

where the numerator is the actual fine-mesh compliance reduction relative to the previously accepted state and the denominator is the coarse-mesh compliance reduction accumulated over the same verification interval. A value $\kappa \approx 1$ indicates strong agreement between the coarse and fine models; values well below 1 signal that the coarse model overestimates the achievable improvement.

The trial design is accepted if and only if both of the following conditions are satisfied:

$$\kappa \;\geq\; \kappa_{\text{tr}} \qquad \text{and} \qquad C_{\text{trial}} \;\leq\; C_{\text{best}} \;-\; \delta_{\text{acc}}\,, \tag{3.32}$$

where κ_{tr} is the trust-ratio threshold, σ_{ratio} is a chosen margin ratio and $\delta_{\text{acc}} = \sigma_{\text{ratio}}|C_{\text{best}}|$ enforces a minimum verified improvement over the best design. The first condition ensures that the coarse model is a reliable predictor, directly adapting the classical

trust-region acceptance rule (Eq. 3.30) to the present setting. The second condition enforces that the trial design must actually improve upon the best verified design by a margin proportional to $|C_{\text{best}}|$, preventing the acceptance of stagnating designs. The specific parameter values (κ_{tr}, σ_{ratio}) are provided in the parameter tables of Chapter 4 (Section 4.1.3). Algorithm 1 summarizes the MFAS loop for two-dimensional implementation.

Algorithm 1 MFAS optimization loop — 2D implementation

Require: $\boldsymbol{\rho}^c$, f_{verify}, κ_{tr}, σ_{ratio}, $b_{\max}$, α_{shrink}, $k_{\max}$, $k_{\min}$, $\text{tol}_{\text{change}}$, k_{consec}

$\boldsymbol{\rho}^f_{\text{best}} \leftarrow P(\boldsymbol{\rho}^c)$; fine FEA $\rightarrow C_{\text{best}}$ {▷ initialise fine-mesh state}
$C_{\text{coarse,old}} \leftarrow$ coarse FEA$(\boldsymbol{\rho}^c)$; $C_{\text{prev}} \leftarrow C_{\text{best}}$ {▷ initialise coarse state}
for $k = 1, 2, \ldots, k_{\max}$ **do**
  coarse FEA; OC update $\rightarrow \boldsymbol{\rho}^c_{\text{new}}$; coarse FEA$(\boldsymbol{\rho}^c_{\text{new}}) \rightarrow C_{\text{coarse,trial}}$ {▷ coarse step}
  if $k \bmod f_{\text{verify}} = 0$ **then**
    $\boldsymbol{\rho}^f_{\text{trial}} \leftarrow P(\boldsymbol{\rho}^c_{\text{new}})$; fine FEA $\rightarrow C_{\text{trial}}$ {▷ verification}
    if $|C_{\text{coarse,old}} - C_{\text{coarse,trial}}| \le 10^{-12} \max(1, |C_{\text{prev}}|)$ **then**
      accept $\leftarrow [C_{\text{trial}} \le C_{\text{prev}}]$ {▷ guard case}
    else
      $\kappa \leftarrow (C_{\text{prev}} - C_{\text{trial}})/(C_{\text{coarse,old}} - C_{\text{coarse,trial}})$ {▷ trust ratio}
      accept $\leftarrow [\kappa \ge \kappa_{\text{tr}}]$ and $[C_{\text{trial}} \le C_{\text{best}} - \sigma_{\text{ratio}}|C_{\text{best}}|]$ {▷ acceptance test}
    end if
    if $\neg$ accept **then**
      attempt $b_{\max}$ backtracked candidates; if any accepted set accept $\leftarrow$ true; else $\boldsymbol{\rho}^c \leftarrow R(\boldsymbol{\rho}^f_{\text{best}})$, $C_{\text{prev}} \leftarrow C_{\text{best}}$ {▷ backtrack or resync}
    end if
    if accept **then**
      $\boldsymbol{\rho}^c \leftarrow \boldsymbol{\rho}^c_{\text{new}}$; $C_{\text{prev}} \leftarrow C_{\text{trial}}$; update $\boldsymbol{\rho}^f_{\text{best}}$, C_{best} if improved {▷ update state}
    end if
    $C_{\text{coarse,old}} \leftarrow C_{\text{coarse,trial}}$
  else
    $\boldsymbol{\rho}^c \leftarrow \boldsymbol{\rho}^c_{\text{new}}$ {▷ non-verification step}
  end if
  if density change $< \text{tol}_{\text{change}}$ for k_{consec} iters and $k \ge k_{\min}$ **then**
    break {▷ convergence}
  end if
end for
return $\boldsymbol{\rho}^f_{\text{best}}$, C_{best}

In the three-dimensional implementation the acceptance criterion is simplified to a

single relative-tolerance condition:

$$C_{\text{trial}} \;\leq\; \big(1+\epsilon_{\text{rel}}\big)\, C_{\text{best}}\,. \tag{3.33}$$

The trial design is accepted whenever its fine-mesh compliance does not exceed the best verified compliance by more than a fraction ϵ_{rel}. If, additionally, $C_{\text{trial}} < C_{\text{best}}$, the best state is updated. This form requires only the scalar state C_{best} and does not involve coarse-compliance tracking, making it straightforward to implement in the GPU-based three-dimensional solver where the PCG solution state must also be managed. The tolerance ϵ_{rel} serves a role analogous to κ_{tr} and σ_{ratio} jointly: it controls the trade-off between accepting marginal proposals that may yield exploration benefit and rejecting proposals that degrade solution quality. Algorithm 2 summarizes the MFAS loop for three-dimensional implementation.

Algorithm 2 MFAS optimization loop — 3D implementation

Require: $\boldsymbol{\rho}^c$, f_{verify}, ϵ_{rel}, n_{stag}, $n_{\text{ver,min}}$, $k_{\max}$

$\boldsymbol{\rho}^f_{\text{best}} \leftarrow P(\boldsymbol{\rho}^c)$; fine FEA $\rightarrow C_{\text{best}}$, $\mathbf{u}_{\text{best}}$ {▷ initialise fine-mesh state}
$n_{\text{ver}} \leftarrow 0$; $n_{\text{no-imp}} \leftarrow 0$
for $k = 1, 2, \ldots, k_{\max}$ **do**
  coarse FEA; OC update $\rightarrow \boldsymbol{\rho}^c_{\text{new}}$ {▷ coarse step}
  if $(k+1) \bmod f_{\text{verify}} = 0$ **then**
    $\boldsymbol{\rho}^f_{\text{trial}} \leftarrow P(\boldsymbol{\rho}^c_{\text{new}})$; fine FEA $\rightarrow C_{\text{trial}}$, $\mathbf{u}_f$; n_{ver} += 1 {▷ verification}
    if $C_{\text{trial}} \leq (1+\epsilon_{\text{rel}})\, C_{\text{best}}$ **then**
      $\boldsymbol{\rho}^c \leftarrow \boldsymbol{\rho}^c_{\text{new}}$ {▷ accept}
      if $C_{\text{trial}} < C_{\text{best}}$ **then**
        $\boldsymbol{\rho}^f_{\text{best}} \leftarrow \boldsymbol{\rho}^f_{\text{trial}}$; $\mathbf{u}_{\text{best}} \leftarrow \mathbf{u}_f$; $C_{\text{best}} \leftarrow C_{\text{trial}}$; $n_{\text{no-imp}} \leftarrow 0$ {▷ update best}
      else
        $n_{\text{no-imp}}$ += 1 {▷ accepted, no improvement}
      end if
    else
      $\boldsymbol{\rho}^f \leftarrow \boldsymbol{\rho}^f_{\text{best}}$; $\mathbf{u}_f \leftarrow \mathbf{u}_{\text{best}}$; $n_{\text{no-imp}}$ += 1 {▷ reject, restore fine state}
    end if
    if $n_{\text{ver}} \geq n_{\text{ver,min}}$ and $n_{\text{no-imp}} \geq n_{\text{stag}}$ **then**
      break {▷ early stop}
    end if
  else
    $\boldsymbol{\rho}^c \leftarrow \boldsymbol{\rho}^c_{\text{new}}$ {▷ non-verification step}
  end if
end for
return $\boldsymbol{\rho}^f_{\text{best}}$, C_{best}

The three-dimensional implementation follows the same high-level MFAS structure as the two-dimensional variant, but it introduces two modifications motivated by the larger fine–coarse discretization gap and the higher cost of fine-mesh PCG solves. First, verification is performed at the end of each f_{verify}-iteration block. Second, the two-dimensional dual acceptance test is replaced by the single relative-tolerance criterion in (Eq. 3.33). This simplification avoids explicit coarse-model agreement tracking while retaining periodic fine-mesh verification as the governing safeguard.

The two-dimensional criterion in (Eq. 3.32) explicitly combines model-agreement testing with a strict verified-improvement requirement. By contrast, the three-dimensional criterion in (Eq. 3.33) accepts a proposal whenever its verified fine-mesh compliance remains within a prescribed relative tolerance of the current best verified state. In both cases, the purpose is the same: to prevent persistent coarse-model drift and to ensure that the design returned by the MFAS loop remains anchored to fine-mesh-verified physics.

3.5 U-Net Architecture and Integration Pipeline

This section describes the convolutional neural network used to generate an initial density field for the coarse mesh in method variant M4. The network follows the encoder–decoder architecture U-Net introduced by Ronneberger et al. [130]. A critical design decision is that the U-Net serves exclusively as a *warm-start* provider: it supplies an informed initial guess to the coarse-mesh optimizer, but does not replace any physics-based component of the optimization pipeline. All density fields produced by the U-Net are subsequently processed by the MFAS verification loop (Section 3.4) and, where applicable, by fine-mesh cleanup iterations, so that the final design is evaluated and refined against the governing equilibrium equations on the fine mesh before being reported.

3.5.1 Network Inputs and Outputs

The U-Net maps a multi-channel description of the boundary-value problem to a scalar density field, both defined on the *coarse* mesh.

For two-dimensional problems, the input tensor has four channels, each of size $n_y^c \times n_x^c$:

1. *Domain mask* — a binary field that is unity for all designable elements; constant at 1 across the entire mesh in two-dimensional problems, where no passive regions are present.

2. *Load-x field* — a sparse field with the x-component of the applied force at the single coarse element nearest to the load-application node; zero everywhere else.

3. *Load-y field* — analogous field for the y-component of the applied force.

4. *Boundary-condition mask* — a binary field that is unity at elements adjacent to fully fixed boundary edges (left face for cantilever problems); for the MBB beam, additionally unity at the single bottom-right roller-support element.

The output is a single-channel field of size $n_y^c \times n_x^c$ representing the predicted element density ρ_e on the coarse mesh.

For three-dimensional problems, the input tensor has six channels, each of size $n_x^c \times n_y^c \times n_z^c$:

1. *Domain mask* — a binary field that is unity inside the active design domain and zero in passive (void) regions.

2. *Fixed-DOF mask* — a binary field marking elements adjacent to Dirichlet boundary faces.

3. *Load-x field* — a scalar field encoding the x-component of the applied force at the loaded elements.

4. *Load-y field* — analogous for the y-component.

5. *Load-z field* — analogous for the z-component.

6. *Volume-fraction field* — a spatially uniform field filled with the prescribed volume fraction v_f.

The output is a single-channel field of size $n_x^c \times n_y^c \times n_z^c$. In both cases the network output passes through a sigmoid activation, so that every predicted element density lies in the interval $(0, 1)$. The post-processing steps that enforce physical admissibility are described in Section 3.5.4.

3.5.2 Architecture

The U-Net architecture adopted in this thesis is a shallow variant of the original design proposed by Ronneberger et al. [130]. The original architecture comprises four encoder levels and four decoder levels with a doubling of feature channels at each downsampling step; the implementation used here reduces the depth to two encoder levels and two decoder levels plus a bottleneck, which is sufficient for the moderate spatial resolutions of the coarse meshes employed (Section 3.1.2).

Each encoder level consists of two successive 3×3 (or 3×3×3 in three dimensions) convolutions with same-padding, each followed by a nonlinear activation, and a 2×2 (or 2×2×2) max-pooling operation with stride 2 for spatial downsampling. Feature channels are doubled at each level. A single convolutional block—two convolutions

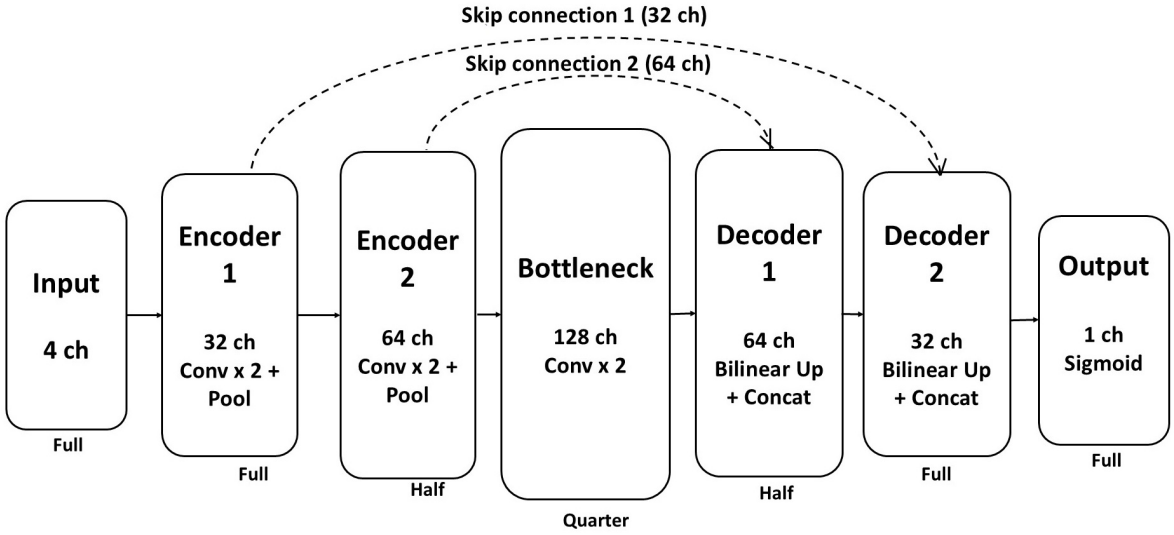

Figure 3.3: Architecture of the 2D U-Net used for coarse-density prediction.

with activation—operates at the coarsest spatial resolution, producing the highest-dimensional feature representation. In the two-dimensional variant, each decoder level upsamples via bilinear interpolation (factor 2×2). In the three-dimensional variant, a 2×2×2 transposed convolution is used instead. This is followed by a concatenation with the correspondingly sized feature map from the encoder (*skip connection*), and two 3×3 (or 3×3×3) convolutions with activation. The skip connections allow the decoder to recover fine-grained spatial information from the encoder, which is essential for accurate density localization. The U-Net architecture adopted in this study has been illustrated in Figure 3.3.

A 1×1 (or 1×1×1) convolution maps the final feature map to a single output channel, and a sigmoid activation constrains the output to the interval $(0, 1)$. In the two-dimensional variant, Rectified Linear Unit (ReLU) activations [131] are used throughout and no batch normalization is applied. In the three-dimensional variant, each convolution is followed by batch normalization [132] and then a ReLU activation. The inclusion of batch normalization in the three-dimensional case stabilizes training given the smaller dataset and batch size available for the three-dimensional benchmarks (Section 4.1.3).

In the three-dimensional variant, the transposed-convolution upsampling may produce spatial dimensions that do not exactly match those of the corresponding encoder feature map when the input dimensions are not powers of two. When a mismatch is detected, the decoder feature map is resized to the encoder dimensions via trilinear interpolation before the skip-connection concatenation is performed. Table 3.1 summarizes the architectural parameters for both variants.

Table 3.1: U-Net architectural parameters for the two-dimensional and three-dimensional implementations.

Parameter	**2D**	**3D**
Input channels	4	6
Output channels	1	1
Encoder levels	2	2
Decoder levels	2	2
Base feature channels	32	16
Batch normalisation	No	Yes
Activation	ReLU	ReLU
Output activation	Sigmoid	Sigmoid
Convolution kernel	3×3	3×3×3
Upsampling	Bilinear interp. 2×	Trans. conv. 2×2×2
Dimension-mismatch fix	—	Trilinear interp.

3.5.3 Training Objective and Data

Training pairs $(\mathbf{x}_i, \mathbf{y}_i)$ are obtained by running the coarse-mesh SIMP optimizer (Section 3.2) to convergence on a set of problem instances that vary the load location, load magnitude, and the boundary-condition type. The input $\mathbf{x}_i$ is the multi-channel tensor described in Section 3.5.1; the target $\mathbf{y}_i$ is the converged coarse-mesh density field.

The two-dimensional training objective is defined as follows [133]:

$$\mathcal{L}_{\text{2D}} \;=\; 0.7\,\text{MSE}(\hat{\mathbf{y}},\,\mathbf{y}) \;+\; 0.3\,\big(1-\text{SSIM}(\hat{\mathbf{y}},\,\mathbf{y})\big)\,, \tag{3.34}$$

where $\hat{\mathbf{y}}$ denotes the network prediction and $\mathbf{y}$ is the target density field. The MSE term drives element-level accuracy across the full density field, which is essential for the warm-start initializer to produce a physically plausible coarse design. The SSIM term, computed over a sliding 3×3 window with stabilization constants $C_1 = 0.01^2$ and $C_2 = 0.03^2$, acts as a structural regularizer: topology optimization density fields possess connectivity properties—load-path continuity, member boundaries, void regions—that element-wise MSE does not capture. The 0.7/0.3 weighting was determined empirically by evaluating weight pairs $\{(1.0, 0.0), (0.7, 0.3), (0.5, 0.5)\}$ on the held-out validation set; the selected ratio minimized validation loss while preserving topological connectivity of the predicted density field. The dataset is split into training, validation, and test subsets in a ratio of 80/10/10, and the model checkpoint with the lowest validation loss is retained. Training and validation loss tracks closely throughout, with validation SSIM plateauing at approximately 0.61 by epoch 30; Figure 3.4 shows the full training history.

The three-dimensional dataset comprises N_{3D} samples, substantially smaller than the two-dimensional dataset. all samples are used for training with no held-out split. Three considerations motivate pure MSE in this setting. First, volumetric SSIM requires evaluating local mean, variance, and cross-covariance statistics over a three-dimensional

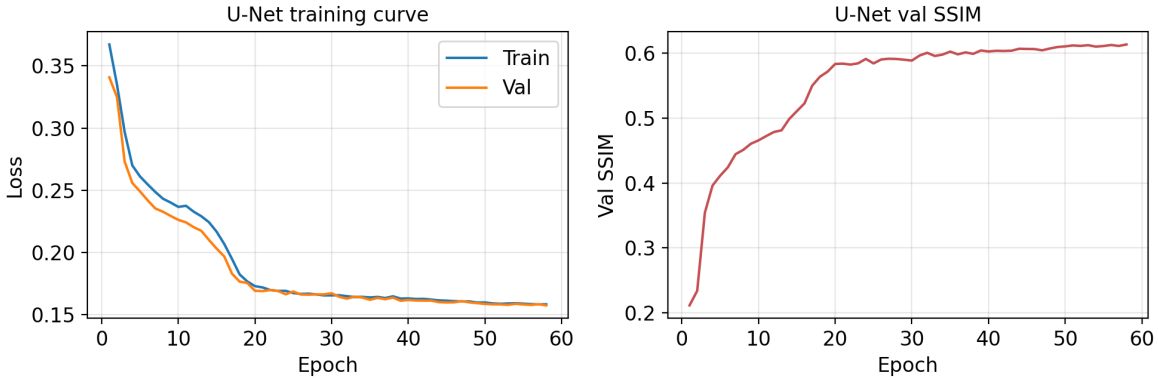

Figure 3.4: Training history of the two-dimensional U-Net. Left: composite training and validation loss over 60 epochs. Right: validation SSIM, which increases monotonically and plateaus at approximately 0.61 by epoch 30, indicating that the network has learned the structural layout of coarse-mesh density fields.

sliding window. For a volume of size $n_x \times n_y \times n_z$, where n_x, n_y, and n_z are the voxel counts in the three spatial directions, and for a cubic window of width w, a direct implementation processes w^3 voxels per window at each of the $n_x n_y n_z$ voxel locations. The computational cost is therefore proportional to $\mathcal{O}(n_x n_y n_z\, w^3)$, making volumetric SSIM significantly more expensive than element-wise MSE for the present 3D setting. Here, $\mathcal{O}(\cdot)$ denotes Big-O notation, that is, the asymptotic order of growth of the computational cost with respect to the problem size. Second, SSIM statistics are sensitive to batch size, with instability increasing for the small batches required by 3D volumes. Third, with limited training data, minimizing hyperparameter count improves generalization stability. The 3D U-Net accordingly serves as a proof-of-concept warm-start initializer; its outputs are subject to the same MFAS fine-mesh acceptance criteria as any coarse-mesh proposal (Section 3.4.3), ensuring warm-start quality is verified rather than assumed. A pure MSE loss is therefore employed:

$$\mathcal{L}_{\text{3D}} \;=\; \text{MSE}(\hat{\mathbf{y}},\, \mathbf{y})\,. \tag{3.35}$$

The L-bracket model converges rapidly and without overfitting, while the cantilever model shows a moderate generalization gap; the full training curves are shown in Figure 3.5. Both variants are trained using the Adam optimizer [134]. The two-dimensional variant uses a learning rate of 5×10^{-4} with weight decay 10^{-5}. The three-dimensional variant uses a learning rate of 10^{-3} with weight decay 10^{-5}. Training proceeds for a maximum of 60 epochs (2D) and 80 epochs (3D).

3.5.4 Post-Processing and Pipeline Injection

The raw network output is a real-valued field in $(0, 1)$ that is not guaranteed to satisfy the prescribed volume fraction or to respect the lower bound $\rho_{\min}$ on element densities.

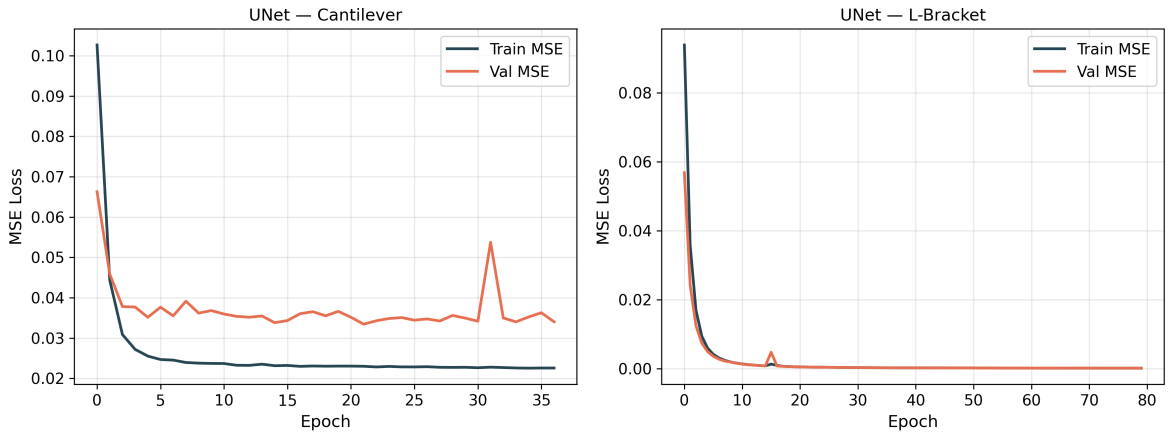

Figure 3.5: Training history of the three-dimensional U-Net for the cantilever (left) and L-bracket (right) problems.

Before the predicted field can serve as an initialization for the coarse-mesh optimizer, the following post-processing steps are applied in sequence:

1. *Clamping.* Every element density is clamped to the interval $[\rho_{\min}, 1]$.

2. *Volume enforcement.* A bisection-based uniform scaling procedure adjusts the density field so that the mean density equals the prescribed volume fraction v_f. Let $\hat{\rho}_e^{\mathrm{NN}} \in (0, 1)$ denote the raw U-Net prediction on the coarse mesh. A scalar multiplier s is then determined by bisection so that the post-processed field satisfies the prescribed average volume fraction over the active coarse-element set $\mathcal{A}_c$:

$$v_f = \frac{1}{|\mathcal{A}_c|} \sum_{e \in \mathcal{A}_c} \mathrm{clamp}\big(s\, \hat{\rho}_e^{\mathrm{NN}},\, \rho_{\min},\, 1\big). \tag{3.36}$$

   For fully designable domains, $\mathcal{A}_c = \{1, \ldots, N_c\}$ and $|\mathcal{A}_c| = N_c$. where N_c is the number of coarse elements (or the number of active elements in domains with passive regions).

3. *Injection.* The post-processed density field is assigned as the initial design-variable vector on the coarse mesh. The coarse-mesh optimizer then continues from this warm start rather than from a uniform initialization at density v_f.

After injection, the coarse-mesh MFAS loop proceeds exactly as described in Section 3.4. Periodic fine-mesh verification ensures that the warm-started design is subject to the same acceptance criteria as any other coarse-mesh proposal. The U-Net prediction is therefore never accepted without physics-based validation; it merely provides a more informed starting point that can accelerate convergence of the coarse optimizer.

3.5.5 Discussion and Limitations

The primary advantage of the U-Net initialization is reduced iteration count for the coarse optimization stage, particularly for problems with strong geometric regularity (e.g., MBB beam and cantilever benchmarks). However, the network is limited by the diversity and coverage of the training dataset. Generalization to unseen load cases or geometry features may degrade, and the predicted density may violate subtle problem-specific constraints unless explicitly encoded in the input. For this reason, the U-Net is used only as an initializer and the final design is always determined by physics-based optimization and fine-mesh verification.

The MFAS additionally prevents the optimization trajectory from drifting into regions where the coarse model is no longer consistent with the fine model. This safeguard is essential when using data-driven initialization, because it ensures that the final design remains verifiably accurate even if the initial guess is imperfect.

3.6 Method Variants

The thesis evaluates four method variants, labeled M1 through M4, that share the finite element model (Section 3.2), the density pipeline (Section 3.3), and the OC update rule (Section 3.3.3), but differ in their use of the coarse mesh, the MFAS verification loop (Section 3.4), and the U-Net initialization (Section 3.5). The four variants are defined in this section and further referenced by label throughout the remainder of the thesis.

3.6.1 M1: Fine-Mesh SIMP Baseline

Method M1 is the reference baseline. The design-variable field is initialized uniformly as $\rho_e = v_f$ for all fine-mesh elements. This field is then passed through the density pipeline to obtain the physical density field $\bar{\rho}_e$ used in the stiffness interpolation and equilibrium solve. At every iteration, the density pipeline, the equilibrium solve $\mathbf{K}(\bar{\boldsymbol{\rho}}_f)\mathbf{u}_f = \mathbf{F}_f$, the compliance sensitivity, and the OC update are evaluated on the fine mesh. The optimizer terminates when the convergence criterion (Eq. 3.22) is met or when the maximum iteration count is reached.

M1 incurs the highest per-iteration cost because every finite element analysis is performed on the fine mesh. It serves as the gold standard against which the compliance quality and the computational cost of M2–M4 are compared: the speedup of any multifidelity variant is measured relative to the wall-clock time of M1 on the same problem instance.

3.6.2 M2: Coarse-Mesh SIMP with Fine Evaluation

Method M2 replaces the iterative optimization mesh with the coarse mesh, thereby exploiting the factor-of-four (2D) or factor-of-eight (3D) reduction in degrees of freedom. Starting from a uniform initialization $\rho_e = v_f$ on the coarse mesh, M2 runs the SIMP optimizer to convergence on the coarse mesh using the coarse-mesh filter radius $r_{\min,c}$. Upon convergence, the coarse-mesh design is transferred to the fine mesh via the prolongation operator P and volume enforcement (Section 3.4), and a single fine-mesh compliance evaluation is performed. No fine-mesh optimization iterations are executed; the fine-mesh compliance is recorded solely for comparison with M1.

M2 provides a lower bound on computational cost (among the four variants) and an upper bound on compliance penalty, because it entirely forgoes fine-mesh optimization. It establishes the baseline cost–quality trade-off against which the MFAS variants (M3, M4) are assessed.

3.6.3 M3: MFAS with Periodic Verification

Method M3 augments the coarse-mesh optimizer with the full MFAS verification loop defined in Section 3.4. The complete MFAS workflow for method variant M3 is demonstrated in Figure 3.6. The cleanup result is reported as the M3 output. The coarse-mesh design variables are initialized uniformly at $\rho_e = v_f$. The initial coarse design is prolongated to the fine mesh via P, volume-enforced, and evaluated to obtain the initial fine-mesh compliance C_{best}. At every iteration, one coarse-mesh SIMP step (filter, sensitivity, OC update) is performed. Every f_{verify} coarse iterations, a *verification step* is triggered:

1. The current coarse design is prolongated to the fine mesh via P and volume-enforced.
2. A fine-mesh compliance evaluation yields C_{trial}.
3. The acceptance criterion (Section 3.4) is applied.
4. If accepted, C_{best} and the stored best fine design are updated accordingly.
5. If rejected, the coarse design is *resynchronized*: the best stored fine design is restricted to the coarse mesh via the restriction operator R, volume-enforced, and assigned as the current coarse state, so that the coarse optimizer resumes from the most recent verified design.

Between verification points, the coarse optimizer runs without fine-mesh involvement. The three-dimensional MFAS loop does not apply the density-change convergence criterion (Eq. 3.22) and termination is governed exclusively by the early-stopping rule

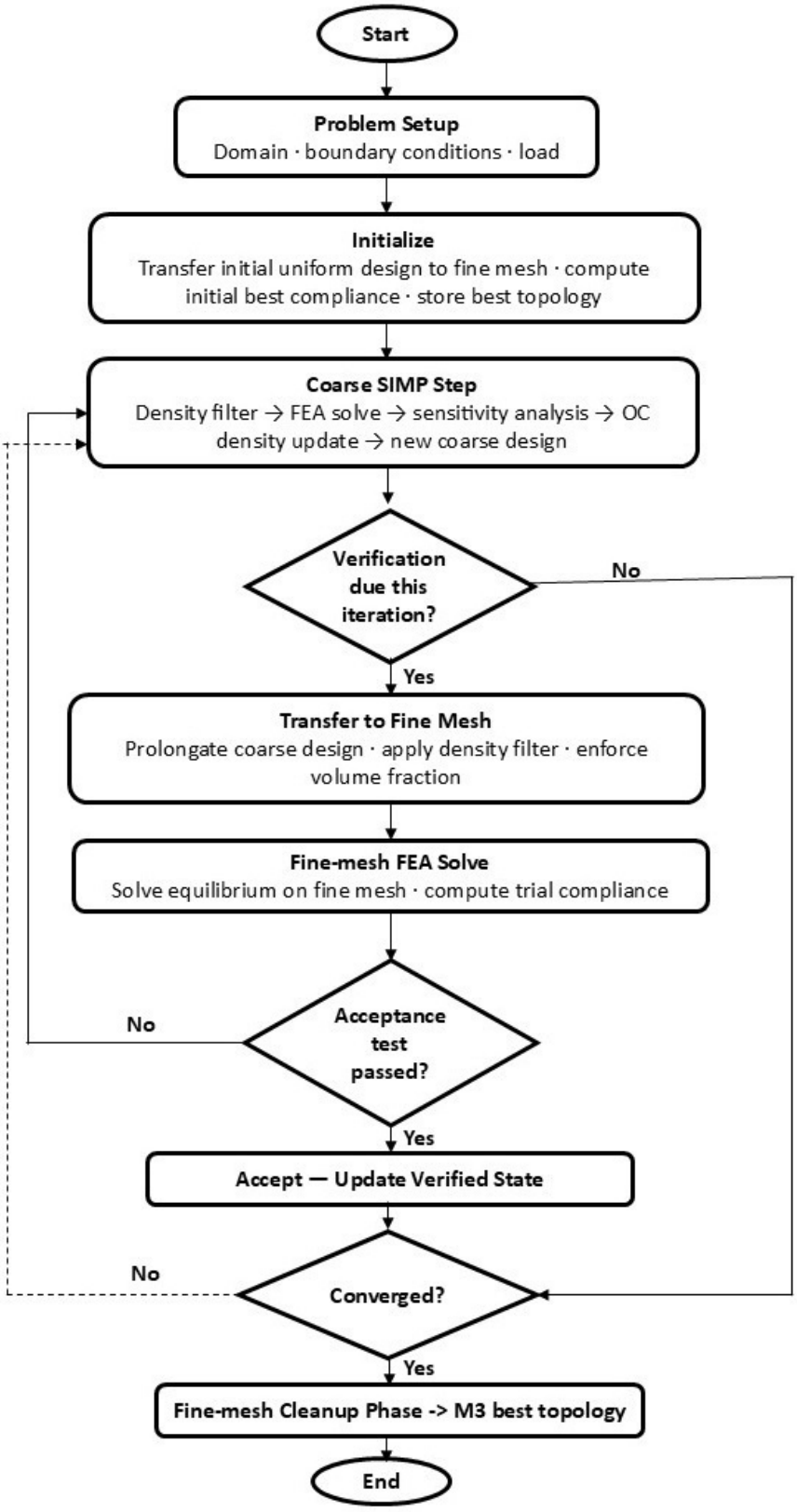

Figure 3.6: Flowchart of the proposed multifidelity acceptance-safeguard optimization loop for method variant M3.

described in Section 3.4.2. Acceptance follows the two-dimensional trust-ratio criterion (Eq. 3.32) and the three-dimensional relative-tolerance criterion (Eq. 3.33), respectively, as defined in Section 3.4.3.

In the three-dimensional implementation, M3 terminates early if no improvement to C_{best} has occurred for four consecutive verification cycles, provided that at least ten verifications have been performed. This criterion prevents continued coarse iterations after the design has stagnated. Upon termination, the stored best fine-mesh design (with compliance C_{best}) is reported as the M3 result.

After the MFAS loop terminates, M3 executes a fine-mesh cleanup phase of N_{cleanup} iterations starting from the final accepted fine-mesh design returned by the MFAS loop. The cleanup applies the fine-mesh filter and OC update; convergence follows the same density-change criterion as the main loop (Eq. 3.22).

3.6.4 M4: MFAS with U-Net Initialization and Fine Cleanup

Method M4 extends M3 with two additional stages: (i) a U-Net–based warm start for the coarse optimizer, and (ii) a fine-mesh cleanup phase that refines the best verified design. The complete M4 pipeline consists of three stages and is demonstrated in Figure 3.7.

Stage 0: U-Net initialization and quality gate. The boundary-condition encoding described in Section 3.5.1 is assembled on the coarse mesh and passed through the trained U-Net. The resulting density prediction is post-processed (clamp, volume enforcement) as described in Section 3.5.4 and assigned as the initial coarse-mesh design. The initial coarse design is prolongated to the fine mesh via P, volume-enforced, and evaluated to obtain the initial C_{best}.

Stage 1: MFAS. The coarse optimizer with periodic verification proceeds exactly as in M3 (Section 3.6.3), including the same acceptance criterion, the resynchronization mechanism on rejection, and (in three dimensions) the early-stopping rules. The only difference from M3 is the starting point: the coarse design begins from the U-Net prediction rather than from a uniform field.

Stage 2: Fine cleanup. After the MFAS phase terminates, N_{cleanup} additional SIMP iterations are performed on the *fine mesh*. In both the two-dimensional and three-dimensional implementations, the cleanup phase starts from the final accepted fine-mesh design returned by the MFAS loop in stage 1; no uniform re-initialization is performed. The fine-mesh filter and OC update are applied throughout. In the two-dimensional implementation, the OC move limit during cleanup is $m = 0.2$, the same value used in the main MFAS loop. In the three-dimensional implementation, the

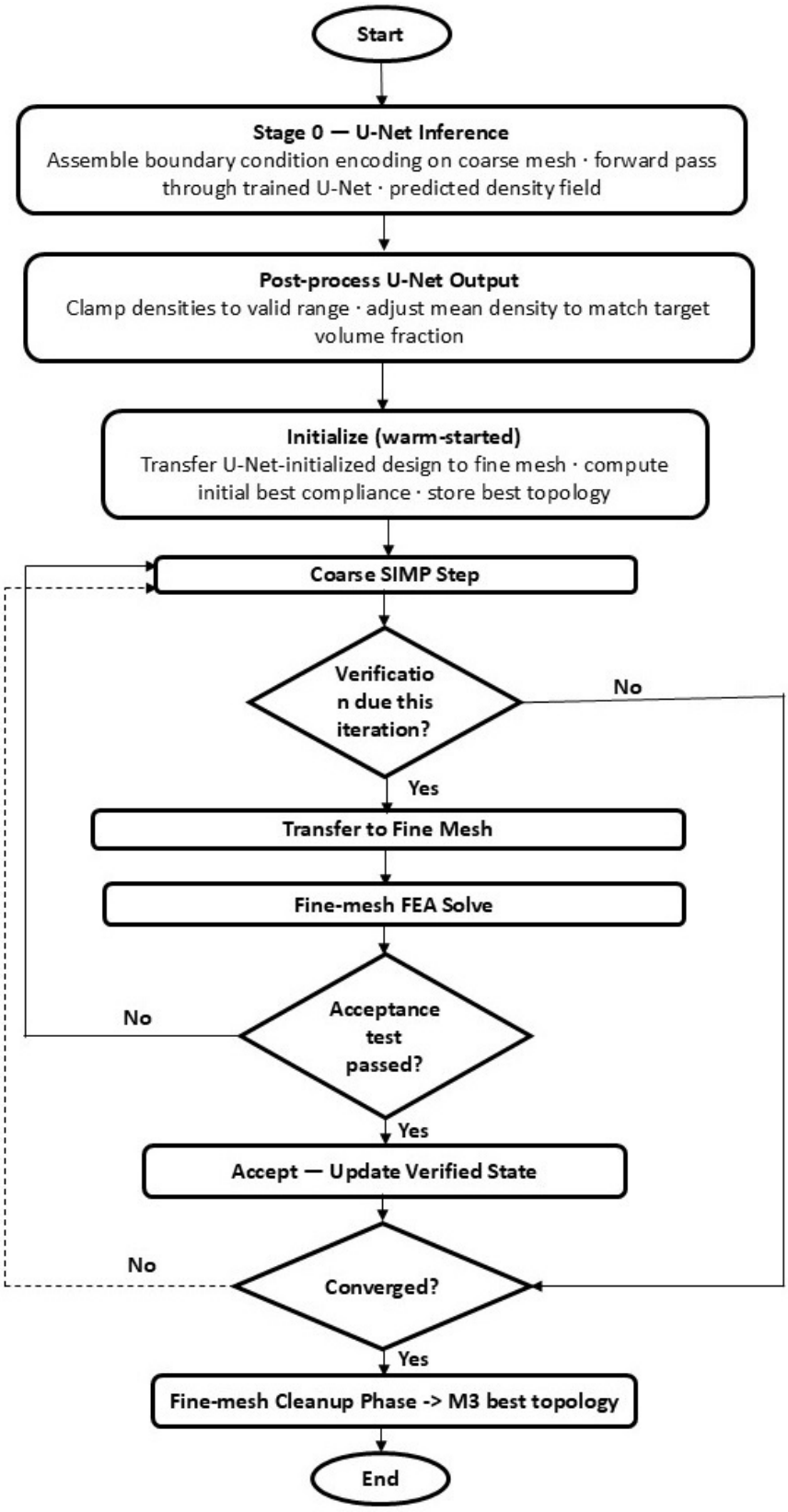

Figure 3.7: Flowchart of the proposed multifidelity acceptance-safeguard optimization loop with U-Net warm start for method variant M4.

move limit is explicitly set to the smallest value in the move-limit schedule (Eq. 3.21). Convergence during cleanup follows the same density-change criterion as the main loop (Eq. 3.22), and the cleanup phase may terminate before N_{cleanup} iterations if convergence is reached.

The density field at the end of the cleanup phase is reported as the M4 result. Its fine-mesh compliance incorporates contributions from all three stages: the U-Net initialization provides structural layout, the MFAS loop corrects and improves the design under physics-based verification, and the fine cleanup resolves remaining gray elements and local suboptimality.

3.6.5 Summary of Method Variants

Table 3.2 provides a compact comparison of the four method variants. The computational cost increases from M2 (cheapest) to M1 (most expensive). M3 is the primary validated contribution, delivering verified acceleration at comparable compliance quality. M4 is the ML-assisted extension: it tests whether a U-Net warm start can further improve initialization quality and acceptance behavior within the same verified loop.

Table 3.2: Summary of method variants M1–M4. *Coarse opt.*: iterative optimization on the coarse mesh. *Fine opt.*: optimization or evaluation on the fine mesh. *MFAS*: multifidelity verification loop active. *U-Net*: network warm-start used. M3 is the primary validated contribution; M4 is the verified ML-assisted variant under evaluation.

Variant	**Coarse opt.**	**MFAS**	**U-Net**	**Fine opt.**	**Purpose**
M1	—	—	—	Full SIMP loop	Fine-mesh baseline
M2	Full SIMP loop	—	—	Eval. only	Cost lower bound
M3	Full SIMP loop	✓	—	Verif. only	Primary MFAS contribution
M4	Full SIMP loop	✓	✓	Cleanup (N_{cleanup} iters.)	ML-assisted MFAS variant

Chapter 4

Numerical Experiments

4.1 Benchmark Problems and Discretizations

This section specifies the benchmark problem geometries, boundary conditions, and loading, then summarizes the numerical parameter values that Chapter 3 explicitly defers to this chapter.

4.1.1 Two-Dimensional Benchmarks

Three canonical 2D compliance-minimization problems are studied on a fine mesh of 180×60 bilinear Q4 elements and a coarse mesh of 90×30 elements, enforcing the strict $2\times$ refinement ratio required by the operators of Section 3.4.1. Table 4.1 summarizes the problem specifications. The three problems are shown in Figure 4.1 and also described below.

Table 4.1: 2D benchmark specifications. Node indices (i_x, i_y): $i_y = 0$ at the bottom edge. All problems share fine mesh 180×60 and coarse mesh 90×30. MBB: left column $u_x = 0$ (symmetry BC), bottom-right corner $u_y = 0$ (roller). Cantilevers: left column fully clamped ($u_x = u_y = 0$).

Problem	v_f	**Load node** (i_x, i_y)	**Load**
MBB beam	0.40	(0, 60)	$F_y = -1$
Tip cantilever	0.50	(180, 0)	$F_y = -1$
Mid cantilever	0.50	(180, 30)	$F_y = -1$

The first problem is the Messerschmitt–Bölkow–Blohm (MBB) beam, solved in its full-domain variant, which corresponds to exploiting the left–right symmetry of the classical half-domain formulation. All nodes on the left column have only their *horizontal* displacement component fixed ($u_x = 0$), enforcing the symmetry condition: vertical motion is unconstrained along this edge. The vertical displacement of the bottom-right corner node is additionally constrained to act as a roller support ($u_y = 0$), preventing rigid-body translation while permitting horizontal sliding. A unit downward

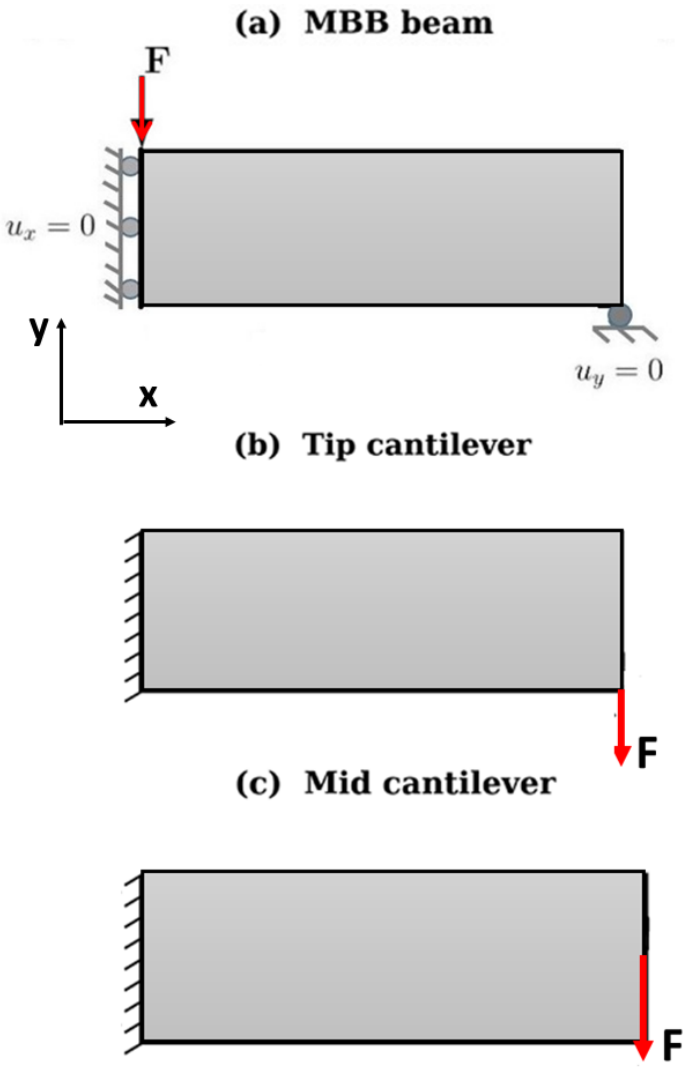

Figure 4.1: Two-dimensional benchmark problems. (a) MBB beam (180×60 elements, $v_f = 0.40$), (b) Tip cantilever (180×60, $v_f = 0.50$), (c) Mid cantilever (180×60, $v_f = 0.50$)

concentrated force of $F_y = -1$ is applied at node $(i_x, i_y) = (0, 60)$, located at the top-left corner of the domain; since only the horizontal DOF of that node is fixed, the applied vertical force acts on a free degree of freedom and produces a well-posed load case. The prescribed volume fraction is $v_f = 0.40$.

The second and third problems are cantilever beams, both discretised on the same 180×60 fine mesh with a fully clamped left column. They differ only in the position of the applied unit downward load $F_y = -1$ and the prescribed volume fraction. In the tip cantilever the load acts at the bottom-right corner, node $(180, 0)$, and the volume fraction is $v_f = 0.50$. In the mid cantilever the load is shifted to mid-height of the right edge, node $(180, 30)$, with the same volume fraction of $v_f = 0.50$. Together, the three problems cover a range of loading eccentricities and volume constraints representative of the standard 2D benchmark suite.

4.1.2 Three-Dimensional Benchmarks

Two 3D benchmarks are studied with trilinear H8 elements; the coarse mesh halves each spatial dimension. Figure 4.2 illustrates the three-dimensional benchmarks considered in this dissertation. Table 4.2 summarizes the specifications.

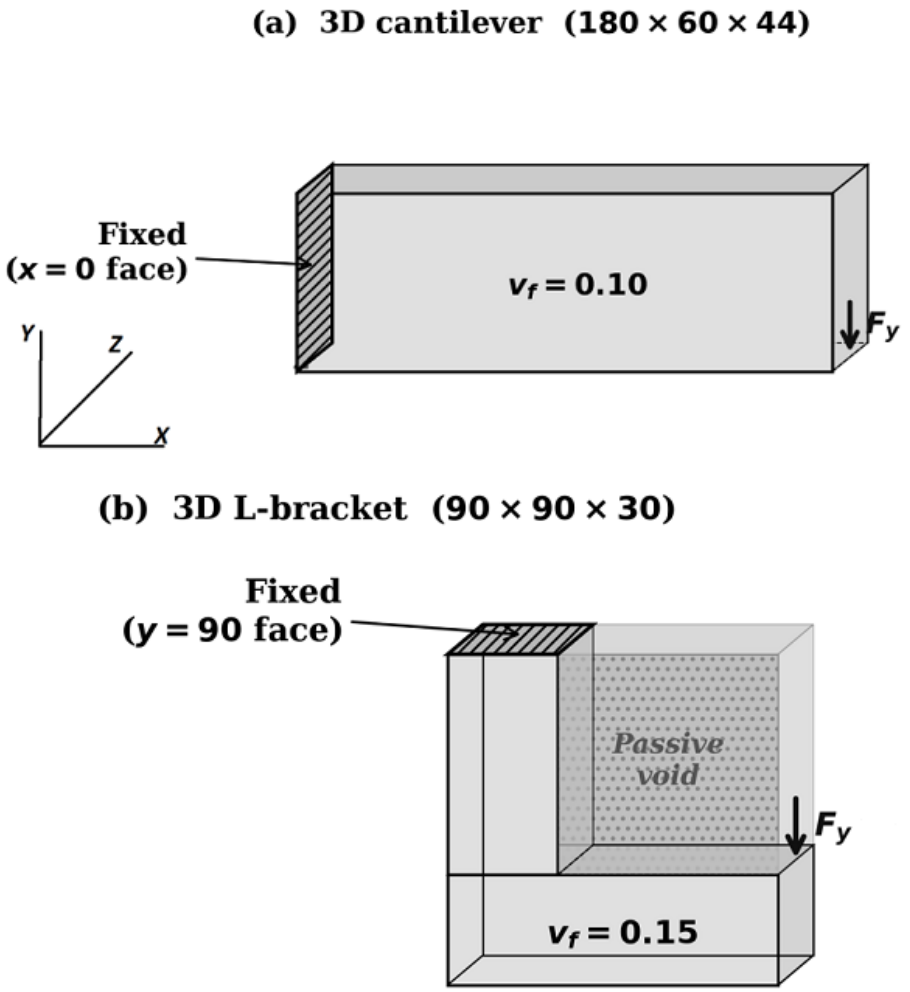

Figure 4.2: Three-dimensional benchmark problems (a) Cantilever ($180 \times 60 \times 44$ elements, $v_f = 0.10$), (b) L-bracket ($90 \times 90 \times 30$ elements, $v_f = 0.15$). The dotted region denotes the prescribed passive void ($i_x \geq n_x/3$ and $i_y \geq n_y/3$). Hatched faces denote fully fixed boundaries.

Table 4.2: 3D benchmark specifications. Cantilever load is a 3-node $-y$ distributed patch with seed-controlled z-offset δ_z; L-bracket load is a full-z-extent $-y$ distributed force (seed-independent).

Problem	**Fine mesh**	**Coarse mesh**	**Fixed BCs**	v_f
Cantilever	$180 \times 60 \times 44$	$90 \times 30 \times 22$	$x = 0$ face clamped	0.10
L-bracket	$90 \times 90 \times 30$	$45 \times 45 \times 15$	$y = n_y$ face clamped	0.15

The first 3D problem is a cantilevered beam discretised on a fine mesh of $180 \times 60 \times 44$ elements and a coarse mesh of $90 \times 30 \times 22$ elements. The face at $x = 0$ is fully clamped. Loading is applied as a downward ($-y$) distributed unit force spread uniformly over a

3-node patch centred at $z = \lfloor n_z^f/2 \rfloor + \delta_z$ on the bottom edge ($y = 0$) of the right face ($x = n_x^f$), where the integer offset $\delta_z \in \{-1, 0, +1\}$ is controlled by the experimental seed. This offset shifts the load patch along the z-axis across replicate runs, introducing geometric diversity while preserving physical equivalence of the structural problem. The entire domain is designable, and the target volume fraction is $v_f = 0.10$.

The second 3D problem is an L-bracket discretised on a fine mesh of $90 \times 90 \times 30$ elements and a coarse mesh of $45 \times 45 \times 15$ elements. The full top face ($y = n_y^f$) is fully clamped, and a downward ($-y$) distributed unit force is applied at the bracket tip located at $x = n_x^f$, $y = \lfloor n_y^f/3 \rfloor - 1$, spread uniformly over all z-nodes at that position. Unlike the cantilever, the L-bracket load position is identical across all seeds. A passive void region is imposed wherever $i_x \geq \lfloor n_x/3 \rfloor$ and $i_y \geq \lfloor n_y/3 \rfloor$ simultaneously, for all z-layers, enforcing the geometric re-entrant corner of the L-bracket. Passive elements are fixed to $\rho_{\min}$ after every update as described in Section 3.4.1, and the volume fraction $v_f = 0.15$ is enforced over the active (designable) elements only.

4.1.3 Numerical Parameter Values

Both M3 and M4 use $N_{\text{cleanup}} = 15$ fine-mesh OC iterations in all executed 2D runs. A default value of 25 appears in the function signature of the cleanup helper but is overridden at every call site in the experiment code; 15 is the authoritative executed value for both methods. M4 additionally executes a 20-iteration warm-start phase on the coarse mesh before the MFAS loop begins (`accept_warmup` = 20). This warm-start cost is included in the M4 wall-clock time reported in all comparisons. Parameter values for 2D experiments are listed in Table 4.3.

Table 4.3: Numerical parameter values for all 2D experiments. The 2D engine uses a single-step convergence test; k_{consec} is not applicable.

Parameter	**Symbol**	**Value**	**Unit**
SIMP penalty	p	3	—
Solid Young's modulus	E_0	1	Pa
Void stiffness	$E_{\min}$	10^{-3}	Pa
Poisson's ratio	ν	0.3	—
Density lower bound	$\rho_{\min}$	10^{-3}	—
Fine filter radius	$r_{\min,f}$	1.5	m †
Coarse filter radius	$r_{\min,c}$	1.5	m †
OC move limit	m	0.2	—
Conv. tolerance	$\text{tol}_{\text{change}}$	10^{-3}	—
Min. iterations	$k_{\min}$	80 (MBB); 100 (TIP, MID)	iter.
Max. iterations (all)	$k_{\max}$	140 (MBB); 180 (TIP, MID)	iter.
Noise-floor ratio	σ_{ratio}	10^{-4}	—
Acceptance threshold	κ_{tr}	0.10 (baseline)	—
M4 warmup window	—	20 coarse iters	iter.
M3/M4 cleanup iterations	N_{cleanup}	15	iter.

Note on non-dimensionalized parameters. All material and geometric parameters in Tables 4.3 and 4.4 are stated in physical SI units where applicable, but the benchmark computations are performed in a non-dimensionalized form consistent with the standard SIMP convention [6], [10], [16]. The normalization sets $E_0 = 1\,\mathrm{Pa}$ and element side length $h = 1\,\mathrm{m}$, so that all derived quantities become dimensionless numbers in the computation. Specifically:

- E_0 and $E_{\min}$ carry units of Pa; their ratio $E_{\min}/E_0$ is the dimensionless void-stiffness fraction (10^{-3} in 2D, 10^{-6} in 3D).
- $r_{\min,f}$ and $r_{\min,c}$ carry units of m (†: numerical values are multiples of $h = 1\,\mathrm{m}$, so $r_{\min,f} = 1.5\,\mathrm{m}$ spans 1.5 element widths).
- $\rho_{\min}$ is a dimensionless relative density (volume fraction), bounded in $[0, 1]$; it is not a physical mass density.
- ν, p, m, κ_{tr}, ϵ_{rel}, and $\mathrm{tol}_{\mathrm{change}}$ are dimensionless by definition.

The compliance values reported throughout this thesis carry units of $\mathrm{Pa} \cdot \mathrm{m}^d$ ($d = 2$ or 3), normalized to order 1 by the choices $E_0 = 1\,\mathrm{Pa}$, $h = 1\,\mathrm{m}$, and $F = 1\,\mathrm{N}$.

Table 4.4: Numerical parameter values for all 3D experiments. k_{consec} and $\mathrm{tol}_{\mathrm{change}}$ apply to M1, M2, and dataset-generation runs only. Cleanup uses a hardcoded early-exit window of 3 consecutive steps (not k_{consec}).

Parameter	**Symbol**	**Value**	**Unit**
SIMP penalty	p	3	—
Solid Young's modulus	E_0	1	Pa
Void stiffness	$E_{\min}$	10^{-6}	Pa
Poisson's ratio	ν	0.3	—
Density lower bound	$\rho_{\min}$	10^{-3}	—
Fine filter radius	$r_{\min,f}$	2.0	m †
Coarse filter radius	$r_{\min,c}$	1.5	m †
OC move schedule	m	0.2 / 0.1 / 0.05 / 0.02	—
Conv. tolerance (M1/M2)	$\mathrm{tol}_{\mathrm{change}}$	0.01	—
Consec. window (M1/M2)	k_{consec}	5	iter.
Min. iterations (M1/M2)	$k_{\min}$	20	iter.
Max. iterations M1	$k_{\max}$	300	iter.
Max. iterations M2	$k_{\max}$	100	iter.
Max. iterations M3, M4	$k_{\max}$	200	iter.
Rel. MFAS tolerance	ϵ_{rel}	0.01 (main); ablation: $\{0.005, 0.01, 0.02, 0.05, 0.10\}$	—
Verification freq. (M3)	f_{verify}	5 (both problems)	iter.
Verification freq. (M4)	f_{verify}	5 (both problems)	iter.
M3/M4 cleanup iterations	N_{cleanup}	15	iter.
Cleanup early-exit window	—	3 (hardcoded)	iter.
PCG primary tolerance	—	10^{-6}; max 400 iter.	—
PCG fallback trigger	—	iter = 400 or residual $> 10^{-3}$	—
PCG fallback tolerance	—	10^{-4}; max 1200 iter.	—

Coarse filter radius. The 2D MFAS engine computes the coarse filter radius as $r_{\min,c} = \max(1.0, r_{\min,f}) = 1.5$, applying the fine-mesh radius directly subject to a

minimum of 1.0. The 3D code sets $r_{\min,c} = 1.5$ directly; this choice produces smoother restricted densities and improves PCG solver conditioning on the coarse mesh. In both cases the executed value is $r_{\min,c} = 1.5$, as recorded in Tables 4.3 and 4.4.

2D convergence window. The general criterion of Eq. (3.22) uses a consecutive window of k_{consec} iterations. The 2D engine implements a *single-step* test: convergence is declared when one OC step satisfies $\max_e |\Delta\rho_e| < \text{tol}_{\text{change}}$ with no window accumulation. The k_{consec} parameter therefore has no entry in Table 4.3. The multi-step form applies only to 3D M1, M2, and dataset-generation runs. The 3D MFAS coarse loop (M3/M4) applies no density-change criterion; it exits via the n_{stag} rule or by exhausting $k_{\max}$. Post-MFAS cleanup in 3D uses a hardcoded early-exit window of 3 consecutive steps.

4.2 Dataset Generation

Training data for the U-Net warm-start initialiser (method M4) are generated before the main comparative experiments by running coarse-mesh SIMP to convergence across a controlled set of problem instances. The U-Net architecture, input–output channel definitions, loss functions, and post-processing pipeline are defined in Section 3.5. This section states the dataset sizes, load-perturbation scheme, seed protocol, splits, and training hyperparameters.

4.2.1 Two-Dimensional Dataset

The 2D dataset contains 1000 samples distributed across the three benchmark problems: 400 for the MBB beam, 300 for the mid-cantilever, and 300 for the tip-cantilever. Each sample is generated by running coarse-mesh SIMP on the 90×30 coarse grid (parameters as in Table 4.3) with a load magnitude drawn from Uniform(0.8, 1.2) via numpy library in Python. The coarse OC loop runs for at most 120 iterations under the single-step convergence criterion with $\text{tol}_{\text{change}} = 10^{-3}$.

Table 4.5: 2D dataset composition and train/val/test partition sizes.

Problem	**Total**	**Train**	**Val**	**Test**
MBB beam	400	320	40	40
Tip cantilever	300	240	30	30
Mid cantilever	300	240	30	30
Total	*1000*	*800*	*100*	*100*

Reproducibility is ensured by a deterministic seed formula: the seed for sample i within problem group g is

$$s_{g,i} \;=\; 123 + 10^5 \cdot \text{key}(g) + i\,, \tag{4.1}$$

where key(g) is a stable integer hash of the group name string, making seeds independent of the ordering of problem groups. Within each group, samples are shuffled with seed 123 and partitioned 80 %/10 %/10 % into training, validation, and test subsets independently, consistent with the split ratio stated in Section 3.5.3.

Regarding U-Net training, Chapter 3 (§3.5.3) states the Adam learning rate (5×10^{-4}), weight decay (10^{-5}), and the composite loss function (Eq. 3.34). The hyperparameters used are a maximum of 60 training epochs with early stopping (patience = 10), batch size 16, and a learning-rate scheduler with patience = 5 and reduction factor 0.5. The implementation uses 60 epochs for the 2D U-Net, and that value is authoritative.

After inference, the U-Net density prediction is post-processed following the pipeline of Section 3.5.4. The filter radius applied at this stage is $r_{\min} = \max(1.0,\, r_{\min,f}/2) = \max(1.0,\, 0.75) = 1.0$, since the lower bound of 1.0 is active when $r_{\min,f}/2 < 1.0$. This value differs from the MFAS coarse-mesh filter radius $r_{\min,c} = 1.5$ used during the subsequent optimization loop (Table 4.3).

4.2.2 Three-Dimensional Dataset

The 3D dataset comprises 180 cantilever and 80 L-bracket base samples. Each is generated by running coarse-mesh SIMP (parameters from Table 4.4) for at most 100 OC iterations, with convergence governed by the same criterion used for M1 and M2: $\text{tol}_{\text{change}} = 0.01$, $k_{\text{consec}} = 5$, and $k_{\min} = 20$ (Eq. 3.22). Seeds cycle in round-robin order as $s_i = 1 + (i \bmod 3)$, which maps to load-patch z-offsets $\delta_z \in \{-1, 0, +1\}$ on the fine cantilever mesh. The corresponding coarse offset is obtained by Python floor division of the fine offset by 2. For the L-bracket the load position does not depend on δ_z, so the cycling seed solely controls the initial random state of the OC solver.

Dataset diversity is doubled through z-axis flip augmentation. Every base sample is augmented by flipping all spatial dimensions along the z-axis. Channel 4 of the 6-channel input tensor (the z-force component F_z, as defined in Section 3.5.1) is negated after flipping in order to maintain physical consistency of the force direction. The target density field is flipped correspondingly. Augmented samples are added to the training partition only and are excluded from validation.

After augmentation, the effective dataset contains 360 cantilever samples and 160 L-bracket samples. The train/validation split is applied to the base samples alone, with val_fraction = 0.20, yielding validation sets of 36 cantilever and 16 L-bracket base samples. Augmented copies of the remaining training-set base samples are then appended to each training partition. There is no test partition for the 3D case. Separate U-Net models are trained for the cantilever and L-bracket problems and are not used interchangeably. Table 4.6 details the resulting partition sizes.

Regarding U-Net training, Chapter 3 (§3.5.3) states the Adam learning rate (10^{-3}),

Table 4.6: 3D dataset composition. Augmented (z-flip) samples enter training only. Val fraction = 0.20 applied to base samples; no test set.

Problem	Base	Eff.	Base train	Aug train	Total train	Val
Cantilever	180	360	144	144	288	36
L-bracket	80	160	64	64	128	16
Total	*260*	*520*	*208*	*208*	*416*	*52*

weight decay (10^{-5}), and the MSE loss function (Eq. 3.35). The hyperparameters used in implementation are per-problem epoch budgets (cantilever: 150; L-bracket: 80) with early stopping (patience = 15, hardcoded), batch size 2, an ExponentialLR learning-rate scheduler with decay factor $\gamma = 0.95$, and a U-Net base channel width of 16 with 6 input channels. The present study uses 60 epochs for the 2D U-Net, 150 epochs for the 3D cantilever U-Net, and 80 epochs for the 3D L-bracket U-Net. After inference, the predicted density is clamped to $[\rho_{\min}, 1]$, the passive mask is applied (L-bracket only), and the volume fraction is enforced over the active domain before the field is passed to the MFAS engine as the coarse initialization.

Table 4.7: U-Net training hyperparameters for the 2D and 3D U-Net models.

Parameter	2D value	3D value
Max. epochs	60	150 (cant.) / 80 (lb.)
Early-stop patience	10	15
Batch size	16	2
LR scheduler	ReduceLROnPlateau ($p = 5$, $f = 0.5$)	ExponentialLR ($\gamma = 0.95$)
Base channel width	—	16

4.3 Experimental Design

The computational study comprises a main comparative matrix and two families of targeted ablation studies. All runs within a given (problem, seed) cell share an identical load configuration, enabling paired comparison across M1–M4.

4.3.1 Main Study Matrix

The 2D main matrix crosses three benchmark problems (MBB, TIP, MID) against four methods (M1–M4) over five replicate seeds (run_id $\in \{1, \ldots, 5\}$), giving 60 runs in total. The random seed for each run is derived deterministically from a stable hash of the string `"{problem}_{method}_{run_id}"`, which ensures that all four methods receive an identical load configuration on any given (problem, seed) cell and that pairwise comparisons are unconfounded by load variation. Iteration caps are set to $k_{\max} = 140$ for the MBB beam and $k_{\max} = 180$ for TIP and MID, reflecting the larger domains of the cantilever problems. Baseline MFAS parameters for M3 and M4 are as

listed in Table 4.8. A validation gate is applied after run_id = 1 on the MBB beam: the empirical acceptance rates of both M3 and M4 must satisfy $\hat{\kappa} \geq 0.70$ before the remaining 59 runs proceed.

Table 4.8: Main study matrix. Seeds: replicate runs per method. M1 and M2 have no MFAS parameters. Both M3 and M4 use $N_{\text{cleanup}} = 15$ cleanup iterations and $f_{\text{verify}} = 5$ in the 2D main experiment. M4 additionally uses a 20-iteration coarse warm-start phase not listed here.

Problem	**Method**	**Seeds**	f_{verify}	**Acceptance param.**	N_{cleanup}
2D MBB beam	M1	5	—	—	—
2D mid cantilever	M2	5	—	—	—
2D tip cantilever	M3	5	5	$\kappa_{\text{tr}} = 0.10$	15
	M4	5	5	$\kappa_{\text{tr}} = 0.10$	15
3D cantilever	M1	3	—	—	—
3D L-bracket	M2	3	—	—	—
	M3	3	5 (both problems)	$\epsilon_{\text{rel}} = 0.01$	15
	M4	3	5 (both problems)	$\epsilon_{\text{rel}} = 0.01$	15
				Total main experiments	*84*

The 3D main matrix crosses two benchmark problems (cantilever, L-bracket) against four methods over three seeds ($s \in \{1, 2, 3\}$), giving 24 runs. For the cantilever, seeds map to load-patch z-offsets $\delta_z \in \{-1, 0, +1\}$, as described in Section 4.1.2. For the L-bracket, the load is identical across all seeds. The seed controls only the initial random state of the solver. Baseline MFAS parameters are problem- and method-specific, as summarised in Table 4.8. Both M3 and M4 use $f_{\text{verify}} = 5$ on all 3D problems. In the 2D main experiment, both M3 and M4 use $f_{\text{verify}} = 5$. The early-stop rule ($n_{\text{stag}} = 4$, $n_{\text{ver,min}} = 10$, stated in §3.4.2) is shared across all 3D MFAS runs. Method M2 in 3D additionally performs a periodic fine-mesh compliance evaluation every $f_{\text{eval}} = 10$ coarse iterations for monitoring purposes. These evaluations are included in the M2 wall-clock time but do not influence the coarse OC update.

4.3.2 Ablation Studies

Two families of targeted ablation studies complement the main matrix. Each ablation varies one parameter while holding all others at the baseline values of Table 4.8. The 2D ablations both use the MBB beam as the benchmark problem and cover run_ids 1–3. The 3D ablations focus on the cantilever problem, with a smaller supplementary study on the L-bracket. Full grids are given in Tables 4.9 and 4.10.

Table 4.9: 2D ablation grids (MBB beam; run_ids 1–3).

Label	**Varied**	**Grid**	**Methods**	**Runs**
Abl A	f_{verify}	$\{3, 5, 10\}$	M3	9
Abl B	κ_{tr}	$\{0.05, 0.10, 0.20\}$	M3, M4	18

Table 4.10: 3D ablation grids. Abl A and B use the cantilever; Abl A2 uses the L-bracket.

Label	Problem	Varied	Grid	Methods	Runs
Abl A	Cantilever	f_{verify}	$\{3, 5, 10\}$	M3	9
Abl B	Cantilever	ϵ_{rel}	$\{0.005, 0.01, 0.02, 0.05, 0.10\}$	M3, M4	20
Abl A2	L-bracket	f_{verify}	$\{3, 10\}$	M3	2

For the 2D case, Ablation A isolates the effect of the verification frequency by sweeping $f_{\text{verify}} \in \{3, 5, 10\}$ for method M3, yielding nine runs in total. Ablation B examines the sensitivity of both M3 and M4 to the trust-ratio acceptance threshold by sweeping $\kappa_{\text{tr}} \in \{0.05, 0.10, 0.20\}$, adding eighteen runs.

For the 3D case, Ablation A sweeps the same verification frequency grid $f_{\text{verify}} \in \{3, 5, 10\}$ for method M3 on the cantilever across all three seeds, contributing nine runs. Ablation B tests the sensitivity of both M3 and M4 to the relative acceptance tolerance by sweeping $\epsilon_{\text{rel}} \in \{0.005, 0.01, 0.02, 0.05, 0.10\}$ on the cantilever across two seeds. The upper end of this grid coincides with the baseline value, so the sweep characterizes the cost of tightening the acceptance criterion. This ablation contributes twenty runs. A smaller supplementary study, Ablation A2, repeats the f_{verify} sweep for M3 on the L-bracket with a reduced grid $\{3, 10\}$ and a single seed, contributing two additional runs.

Across both dimensions, the total count is 87 runs for 2D $(60 + 9 + 18)$ and 55 runs for 3D $(24 + 9 + 20 + 2)$, giving a grand total of 142 runs (Table 4.11).

Table 4.11: Grand total of computational runs.

Component	2D	3D
Main matrix	60	24
Ablation A	9	9
Ablation B	18	20
Ablation A2	—	2
Total	**87**	**55**
	Grand total	**142**

4.3.3 Compute Environment

The 2D experiments (87 runs) ran on an NVIDIA Tesla T4 (14.6 GB VRAM, 40 SMs, compute capability 7.5). The 3D experiments (55 runs) ran on an NVIDIA A100-SXM4-40GB (39.5 GB VRAM, 108 SMs, compute capability 8.0). Both used: Python 3.12.12, PyTorch 2.10.0+cu128, CUDA 12.8, NumPy 2.0.2, SciPy 1.16.3, pandas 2.2.2, matplotlib 3.10.0, scikit-learn 1.6.1. 2D FEA uses a CPU sparse direct solver; 3D FEA uses GPU PCG.

Wall-clock time t is defined as follows. In 2D, the timer spans the entire `run_method` call, including U-Net inference (M4), the full MFAS coarse loop, all fine-mesh verifi-

cation solves, and the cleanup stage. In 3D, the timer for all methods includes every computational stage; for M4 this encompasses Stage 0 U-Net inference ($\approx 1.8\,\mathrm{s}$), Stage 1 MFAS loop, any recovery, and Stage 2 cleanup. For M3, M1, and M2 in 3D, the timer starts at the beginning of the optimization loop and stops after cleanup (M3), after the final fine FEA (M2), or after the full fine-mesh OC loop (M1). All within-dimension comparisons of t are consistent under these definitions.

4.4 Ablation Studies

Two hyperparameters exert the most direct influence over the behaviour of the MFAS acceptance logic: the verification frequency f_{verify} and the acceptance threshold, which is expressed as the trust-ratio threshold κ_{tr} in two dimensions and as the relative tolerance ϵ_{rel} in three dimensions. The verification frequency controls how often the fine-mesh oracle is invoked during the coarse optimization loop, while the threshold parameter governs the stringency with which a coarse-predicted improvement must be corroborated at fine resolution before the corresponding design is accepted. The ablation studies discussed in this section systematically vary each parameter in isolation, holding all others at their main-experiment baseline values, and are confined to the parameter grids, benchmark problems, and seed subsets implemented in the executable experiment code.

4.4.1 Two-Dimensional Ablations

Both two-dimensional ablations are conducted on the MBB beam problem alone ($v_f = 0.40$), using the seed subset $\{1, 2, 3\}$ drawn from the main five-seed pool. The MBB beam is selected because it exhibits the widest compliance dynamic range among the three 2D benchmarks, which makes sensitivity to algorithmic hyperparameters most clearly interpretable. The main-experiment baselines for methods M3 and M4 are $f_{\text{verify}} = 5$, $\kappa_{\text{tr}} = 0.10$, and $\sigma_{\text{ratio}} = 10^{-4}$.

The first ablation (Ablation A) varies the verification frequency for method M3, sweeping $f_{\text{verify}} \in \{3, 5, 10\}$ at fixed $\kappa_{\text{tr}} = 0.10$ and producing nine runs in total. Reducing the verification frequency below the baseline increases the number of fine-mesh solves per unit of coarse optimization time, raising the probability that a low-quality coarse iterate is detected and rejected before it can propagate into the accepted design trajectory. Increasing f_{verify} beyond the baseline reduces the frequency of fine-mesh calls, lowering computational overhead at the cost of allowing the coarse mesh to drift further from the fine-mesh optimum between consecutive verification events. The responses measured for this ablation are the final fine-mesh compliance $C_{\text{final,fine}}$, total wall-clock time, and the acceptance rate $n_{\text{accept}}/n_{\text{verify}}$.

Ablation A is conducted for M3 only and varies one parameter at a time. Its results characterize the sensitivity of the MFAS loop to the verification frequency and motivate the design question of whether sparser schedules are beneficial when the coarse trajectory begins from a better-quality starting point. No change to the main-experiment baselines is inferred from Ablation A alone; both methods retain $f_{\text{verify}} = 5$ in the primary 2D comparison.

The second ablation (Ablation B) examines the sensitivity of both M3 and M4 to the acceptance threshold, sweeping $\kappa_{\text{tr}} \in \{0.05, 0.10, 0.20\}$ at the fixed baseline $f_{\text{verify}} = 5$ and producing eighteen runs. A larger threshold demands that the realised fine-mesh compliance reduction constitute a greater fraction of the coarse-predicted reduction before a proposal is accepted, imposing a more conservative quality gate on the coarse-mesh search. A smaller threshold permits acceptance of proposals whose coarse prediction is only partially corroborated at fine resolution. Because the M4 warmup period spanning the first twenty coarse iterations uses a noise-floor criterion independently of κ_{tr}, the threshold sweep isolates its effect exclusively on the main convergence phase. Both methods are included to test whether $\kappa_{\text{tr}} = 0.10$ is a robust operating point. A threshold that is well-chosen should show limited sensitivity to moderate perturbation. The ablation validates this property and identifies any region of the swept range where the threshold materially affects compliance, runtime, or acceptance rate.

4.4.2 Three-Dimensional Ablations

Three ablation studies are conducted in three dimensions. The cantilever serves as the primary sensitivity benchmark for Ablation A and Ablation B, while Ablation A2 extends the verification-frequency sweep to the L-bracket to test the transferability of the trends observed on the cantilever. The main-experiment baselines for all 3D MFAS runs are $\epsilon_{\text{rel}} = 0.01$ and $f_{\text{verify}} = 5$ for both M3 and M4 on both problems. The value $\epsilon_{\text{rel}} = 0.10$ was chosen because it achieves acceptance rates in the range 75–92 % across both benchmark problems in exploratory runs, providing a balance between acceptance selectivity and coarse-mesh efficiency.

Ablation A varies the verification frequency for method M3 on the cantilever, sweeping $f_{\text{verify}} \in \{3, 5, 10\}$ across all three seeds and producing nine runs. The 3D acceptance rule is the single inequality $C_{\text{trial}} \leq (1 + \epsilon_{\text{rel}})\, C_{\text{best}}$, so the role of f_{verify} in three dimensions is to control how frequently this check is invoked and how quickly a rejected design triggers reinstatement of the best-accepted density field on the coarse mesh. At the start of each new verification cycle, the coarse displacement vector is reset to zero to prevent a stale warm-start from degrading PCG convergence following the density restriction step.

Ablation A2 provides a supplementary verification-frequency sweep for M3 on the L-bracket, using a single seed and the grid $\{3, 10\}$, adding two runs. The main-experiment baseline of $f_{\text{verify}} = 5$ lies between the two swept values, so both a tighter and a sparser cadence are evaluated. The L-bracket presents a structurally distinct load path and a prescribed passive void region compared with the cantilever, so this ablation tests whether the sensitivity trends from Ablation A persist across a more geometrically complex problem.

Ablation B varies the relative acceptance tolerance ϵ_{rel} for both M3 and M4 on the cantilever, sweeping $\epsilon_{\text{rel}} \in \{0.005, 0.01, 0.02, 0.05, 0.10\}$ using seeds $\{1, 2\}$ and producing twenty runs. Neither method receives an explicit f_{verify} override in this ablation, so each retains its cantilever-specific baseline ($f_{\text{verify}} = 5$ for both M3 and M4), thereby isolating the effect of the tolerance parameter from any confound due to verification frequency. The swept range extends from a tight value of 0.005, which approaches a strict non-worsening condition in which virtually any compliance increase causes rejection, up to the most permissive value in the tested sweep, $\epsilon_{\text{rel}} = 0.10$. Very small values of ϵ_{rel} are expected to generate high rejection rates during the plateau phase of convergence, where successive coarse iterates differ by less than the tolerance band, potentially causing premature stagnation.

Table 4.12 summarizes all ablation conditions across both dimensionalities. The combined ablation workload is 27 runs in two dimensions and 31 runs in three dimensions.

Table 4.12: Summary of all ablation conditions. Boldface identifies the main-experiment baseline value within each swept grid. All parameters not listed in the *Swept values* column are held at their respective main-experiment baseline values.

Dim.	ID	Swept parameter	Swept values	Methods	Runs
2D	A	f_{verify}	$\{3, \mathbf{5}, 10\}$	M3 only	9
2D	B	κ_{tr}	$\{0.05, \mathbf{0.10}, 0.20\}$	M3, M4	18
3D	A	f_{verify}, cantilever	$\{3, \mathbf{5}, 10\}$	M3 only	9
3D	A2	f_{verify}, L-bracket	$\{\mathbf{3}, 10\}$	M3 only	2
3D	B	ϵ_{rel}, cantilever	$\{0.005, 0.01, 0.02, 0.05, \mathbf{0.10}\}$	M3 ($f_{\text{v}}=5$), M4 ($f_{\text{v}}=5$)	20
				Total	58

4.5 Evaluation Metrics and Statistical Reporting

A unified evaluation framework is applied across all methods, benchmark problems, and dimensionalities to support consistent, quantitative comparison. The metrics are organized into three groups: primary performance indicators that characterize solution quality and computational cost, MFAS diagnostic metrics that record the internal behavior of the acceptance mechanism, and design-quality measures that quantify the binary character of the final topology. All scalar quantities are written into the master

results files at the end of each run and subsequently post-processed to derive relative and aggregate statistics.

4.5.1 Primary Performance Indicators

The definitive measure of structural performance is the fine-mesh compliance evaluated at termination, referred to throughout as C_{final}. For M1, which operates entirely on the fine mesh, this is the compliance recorded at convergence of the OC loop. For M2, the coarse optimization loop runs to convergence and a single fine-mesh solve is then performed on the prolongated design to obtain C_{final}. In the three-dimensional implementation, M2 additionally executes a periodic fine-mesh evaluation every $f_{\text{eval,M2}} = 10$ coarse iterations during the loop for monitoring purposes, but these intermediate solves do not affect the optimization trajectory or the reported result. For M3 and M4, C_{final} is the compliance of the last accepted fine-mesh design after the shared cleanup phase, which consists of ten fine-mesh OC iterations executed at a reduced move limit of $m = 0.02$.

To compare methods across problems with different absolute compliance scales, the performance gap relative to M1 is computed as

$$\Delta_k^{\text{M1}} \;=\; \frac{C_{\text{final},k} - C_{\text{final,M1}}}{C_{\text{final,M1}}} \times 100\,\%, \tag{4.2}$$

where $k \in \{\text{M2, M3, M4}\}$. In two dimensions this ratio is computed per-problem and per-seed and in three dimensions as the normalized compliance $C_k/\overline{C}_{\text{M1}}$.

Computational efficiency is quantified by the speedup relative to M1,

$$S_k \;=\; \frac{t_{\text{M1}}}{t_k}, \tag{4.3}$$

where t denotes total wall-clock time in seconds, measured from method initialization to the end of the fine cleanup stage (or the terminal fine solve for M1 and M2). For M3, this interval encompasses all coarse iterations, all fine verification solves, and the $N_{\text{cleanup}} = 15$ iteration cleanup phase. For M4, the wall-clock time additionally includes the U-Net inference for coarse initialization. The time is further decomposed into an MFAS phase duration and a cleanup phase duration, which are recorded separately in the result dictionary to enable stage-level efficiency analysis. For M2, the total time includes the coarse loop, any periodic intermediate fine evaluations in 3D, and the terminal fine solve.

Wall-clock time conflates hardware effects with algorithmic behavior, so the number of fine-mesh and coarse-mesh linear-system solves is also logged independently for every run as a hardware-agnostic surrogate for computational work. For M3 and M4, the coarse count accumulates standard coarse OC solves, while the fine count accumulates verification checks and cleanup iterations. For M2 in three dimensions, the periodic

monitoring evaluations in addition to the terminal solve is also included.

4.5.2 MFAS Diagnostic Metrics

The acceptance rate summarizes the overall selectivity of the MFAS loop and is defined as

$$\hat{\kappa} \;=\; \frac{n_{\text{accept}}}{n_{\text{accept}} + n_{\text{reject}}}. \tag{4.4}$$

Methods M1 and M2 are assigned $\hat{\kappa} = 1$ by convention, as they do not perform any accept-or-reject decisions.

In two dimensions, the trust ratio at each verification event j is recorded as a detailed diagnostic. It is computed as

$$\kappa_j \;=\; \frac{\Delta C_{\text{actual},j}}{\Delta C_{\text{pred},j}} \;=\; \frac{C_{f,j-1} - C_{f,j}}{C_{c,j-1} - C_{c,j}}, \tag{4.5}$$

When the coarse-predicted change is numerically negligible ($|C_{\text{coarse,old}} - C_{\text{coarse,trial}}| \leq 10^{-12} \max(1, |C_{\text{prev}}|)$), the trust ratio is undefined and the acceptance decision reverts to a direct fine-mesh comparison: the trial is accepted if and only if $C_{\text{trial}} \leq C_{\text{prev}}$ (Eq. 3.28). The σ_{ratio} margin condition does not apply in this guard case. In three dimensions, the acceptance criterion does not use a trust ratio; it tests only whether $C_{\text{trial}} \leq (1 + \epsilon_{\text{rel}})\, C_{\text{best}}$, so no κ_j sequence is produced.

The sequence of best accepted compliance values $\{C_{\text{best},j}\}$ is recorded at each verification event in both two and three dimensions, and provides the primary convergence diagnostic for the MFAS loop. In three dimensions, convergence is additionally monitored through two complementary early-stopping criteria that are applied jointly once at least $n_{\text{stag,min}} = 10$ verifications have been completed. The first criterion terminates the MFAS phase if the count of consecutive non-improving verification events reaches the patience limit $n_{\text{stag}} = 4$. The second criterion terminates the phase if the relative improvement in C_{best} over the preceding n_{stag} verification windows falls below a configured convergence tolerance ζ:

$$\frac{C_{\text{best},j-n_{\text{stag}}} - C_{\text{best},j}}{C_{\text{best},j-n_{\text{stag}}}} \;<\; \zeta. \tag{4.6}$$

Both criteria are applied to M3 and M4, ensuring that the MFAS phase terminates as soon as the design trajectory has effectively plateaued, regardless of whether the patience counter has been exhausted.

4.5.3 Design-Quality Assessment

The binary character of the final density field is quantified by the gray fraction, defined as the proportion of elements whose physical density $\bar{\rho}_e$ lies in the intermediate range (since $\beta = 0$, $\bar{\rho}_e \equiv \tilde{\rho}_e$ throughout this study):

$$g \;=\; \frac{1}{N}\sum_{e=1}^{N} \mathbf{1}(t_{\text{gray}} < \bar{\rho}_e < 1 - t_{\text{gray}})\,, \tag{4.7}$$

where $t_{\text{gray}} = 0.05$ in both two and three dimensions. A design with a small gray fraction is predominantly composed of solid and void elements, indicating that SIMP penalization has been effective in suppressing intermediate densities. An additional relative gray-reduction metric,

$$G_k^{\text{M2}} \;=\; \frac{g_{\text{M2}} - g_k}{g_{\text{M2}}} \times 100\,\%, \tag{4.8}$$

is computed in two dimensions to quantify the improvement in topological clarity relative to the coarse-only baseline.

4.5.4 Statistical Reporting

Primary performance indicators are collected over five paired seeds in two dimensions and three paired seeds in three dimensions, and the statistical procedures differ between the two dimensionalities in the choice of confidence-interval method, test directionality, and effect-size estimator.

For each method pair (M1, k) in the two-dimensional study, the per-seed compliance difference $d_i = C_{\text{final,M1},i} - C_{\text{final},k,i}$ is tested with a two-tailed Wilcoxon signed-rank test ($n = 5$ pairs). The associated effect size is Cohen's d_z, defined as

$$d_z \;=\; \frac{\overline{d}}{\hat{s}_d}, \tag{4.9}$$

where $\overline{d}$ is the mean and $\hat{s}_d$ the Bessel-corrected standard deviation of the difference sequence $\{d_i\}$. Uncertainty on the mean compliance for each method is reported as a bootstrap 95 % confidence interval computed from 10 000 resamples. The significance level is $\alpha = 0.05$ throughout, with p-values reported to four decimal places.

In the three-dimensional study, each method pair (M1, k) and the pair (M3, M4) are tested per problem using a one-tailed Wilcoxon signed-rank test with $n = 3$ seeds, oriented to test whether method k yields strictly lower compliance than the comparator. The 95 % confidence interval on mean compliance is constructed as a Student-t interval with two degrees of freedom. The effect size is Cohen's d computed with a pooled

standard deviation:

$$d = \frac{\overline{C}_k - \overline{C}_{\mathrm{M1}}}{\sqrt{\frac{1}{2}(\hat{s}_k^2 + \hat{s}_{\mathrm{M1}}^2)}}. \tag{4.10}$$

Significance is evaluated at $\alpha = 0.05$. All three-dimensional statistical results are consumed by the figure-generation routines, which overlay significance markers on compliance summary charts using the convention $*$ for $p < 0.05$ and † for $p < 0.10$. Results for each problem are reported separately, with no pooling across the cantilever and L-bracket.

The 2D SIMP/OC pipeline is effectively deterministic once the problem geometry and mesh are fixed. The only seed-to-seed variation arises from the hash-based load perturbation, which draws the load magnitude from a narrow uniform distribution. Compliance outcomes therefore exhibit negligible variability across the five repeated runs in practice. The repeated runs serve primarily to confirm implementation stability and to provide multiple timing samples for reliable wall-clock estimation. Wilcoxon tests and Cohen's d_z are reported for completeness but should not be over-interpreted for compliance: the near-zero compliance spread reflects the deterministic nature of the optimizer, not a statistical conclusion about method equivalence. Runtime comparisons, which reflect genuine run-to-run variability in system load and scheduling, are the statistically informative quantity in the 2D setting.

Chapter 5

Results and Discussion

This chapter presents and interprets the computational results obtained from the experimental protocol. The main two-dimensional results are discussed in Section 5.1, the three-dimensional results are discussed in Section 5.2. Sections 5.3 and 5.4 present the hyperparameter ablation studies for each dimensionality separately. Section 5.5 synthesises findings into practitioner guidance, and Section 5.6 acknowledges the principal limitations.

5.1 Two-Dimensional Results

5.1.1 Problem setup and scope

Three benchmark problems are evaluated: the MBB beam ($v_f = 0.40$), the tip-loaded cantilever (TIP; $v_f = 0.50$), and the mid-span cantilever (MID; $v_f = 0.50$). Each is solved with methods M1–M4 at $n = 5$ paired seeds following the protocol in Chapter 4. All three problems share the same discretization: fine mesh $180 \times 60 = 10{,}800$ bilinear Q4 elements and coarse mesh $90 \times 30 = 2{,}700$ elements, giving a $4:1$ element-count reduction ratio consistent with the strict $2\times$ refinement rule. Baseline MFAS parameters are $f_{\text{verify}} = 5$ and $\kappa_{\text{tr}} = 0.10$ for both M3 and M4; both use $N_{\text{cleanup}} = 15$ fine-mesh OC iterations. Topology evolution grids for the MBB beam, MID cantilever, and TIP cantilever are shown in Figures 5.1, 5.2, and 5.3 respectively.

5.1.2 Convergence behavior

The convergence behavior of all four methods is examined through the relative compliance and relative gray fraction histories recorded at each iteration. Figures 5.4, 5.5, and 5.6 plot these quantities for the MBB beam, MID cantilever, and TIP cantilever respectively.

Methods M1 and M2 produce smooth, monotone descent curves. Both reach the bulk

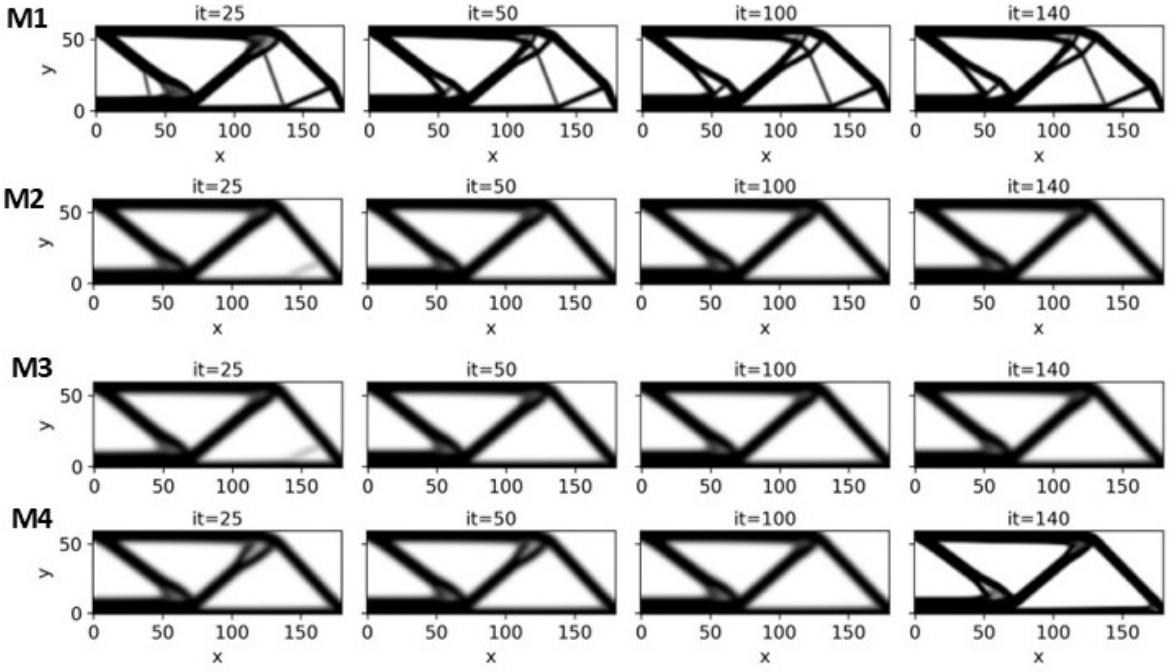

Figure 5.1: Density field snapshots at iterations 25, 50, 100, and 140 for M1–M4 on the MBB beam ($v_f = 0.40$).

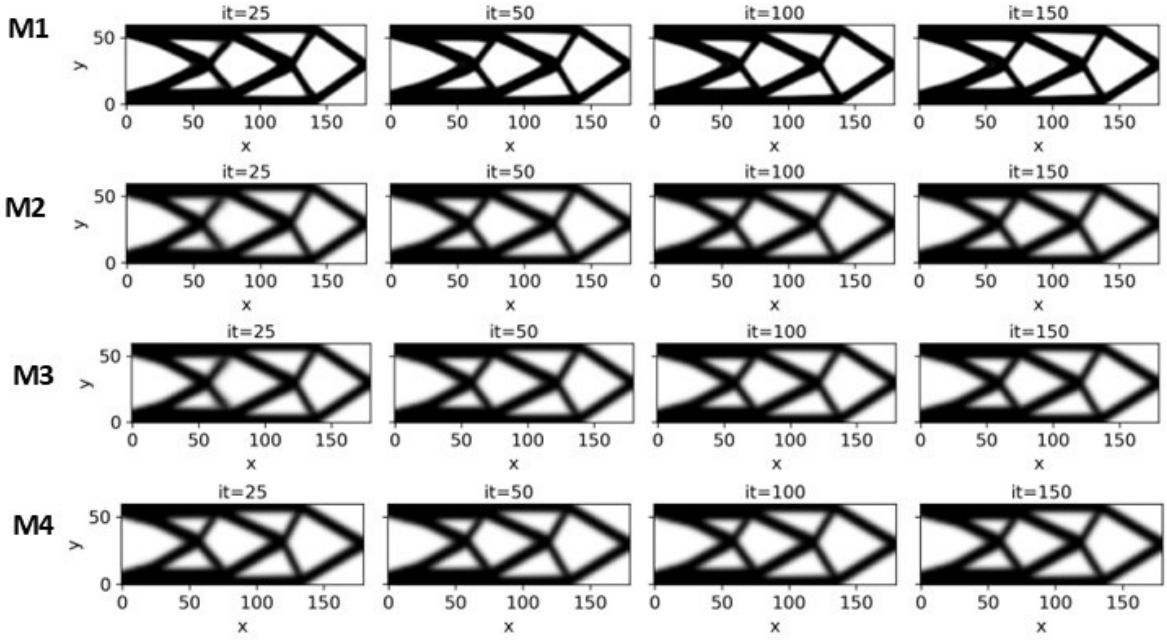

Figure 5.2: Density field snapshots at iterations 25, 50, 100, and 150 for M1–M4 on the MID cantilever ($v_f = 0.50$).

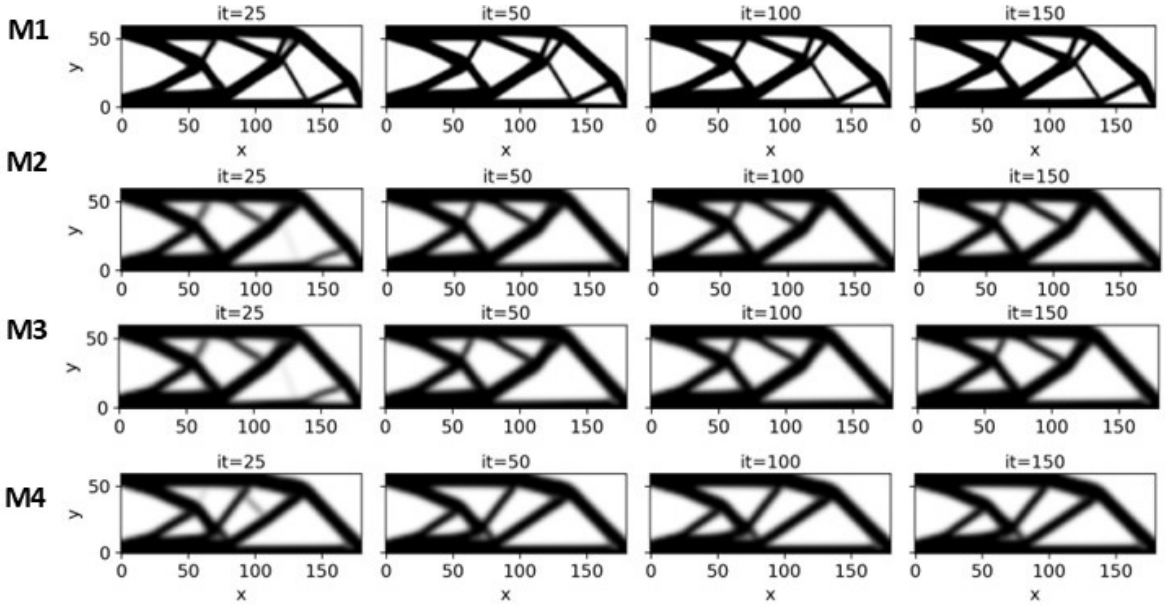

Figure 5.3: Density field snapshots at iterations 25, 50, 100, and 150 for M1–M4 on the TIP cantilever ($v_f = 0.50$).

of their compliance reduction within the first 20–25 iterations, after which the curves flatten into a near-horizontal plateau. Despite their similar compliance trajectories, M2 settles at a substantially higher gray fraction than M1 across all three problems (e.g., $g = 0.344$ versus 0.224 on the MBB beam), a direct consequence of the coarse-to-fine prolongation being applied without any subsequent projection cleanup.

Methods M3 and M4, by contrast, exhibit a characteristic *staircase* descent in both compliance and gray fraction. Each discrete step corresponds to a fine-mesh verification event occurring every $f_{\text{verify}} = 5$ coarse iterations; between verifications, the coarse OC loop advances the density without updating either metric. A sharp, discontinuous drop appears in the final 15 cleanup iterations for both methods, reflecting the post-MFAS fine-mesh SIMP phase that eliminates residual intermediate-density material and drives gray fraction to near-M1 levels.

M4 exhibits prolonged coarse-loop stagnation on the MID and TIP cantilevers. The compliance plateau persists well above the M1-converged level for most of the optimization horizon, with nearly all quality recovery deferred to the cleanup phase. The first fine-mesh verification compliance for M4 is lower than for M3 on every problem (e.g., 327 versus 489 on the MBB beam), confirming that the U-Net warm-start positions the coarse density closer to the fine-mesh attractor early in the run. Nevertheless, on the asymmetric cantilever load cases, the warm-started density field anchors the coarse loop in a basin that diverges progressively from the fine-mesh optimum, explaining the small but nonzero compliance gap of +0.07 % and +0.10 % reported in Table 5.1 for M3 and M4 on the TIP cantilever.

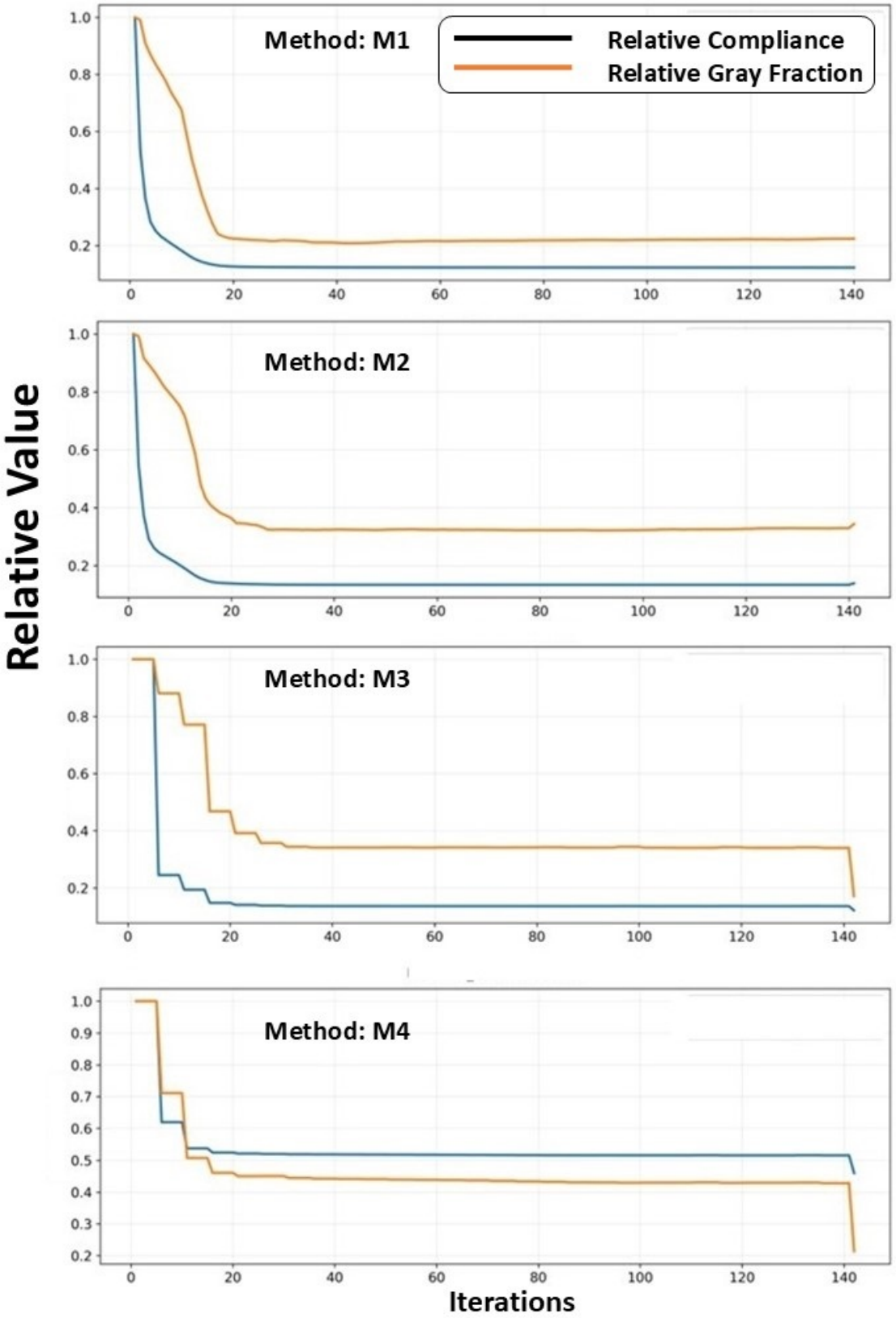

Figure 5.4: Relative compliance and relative gray fraction versus iteration for M1–M4 on the MBB beam ($v_f = 0.40$).

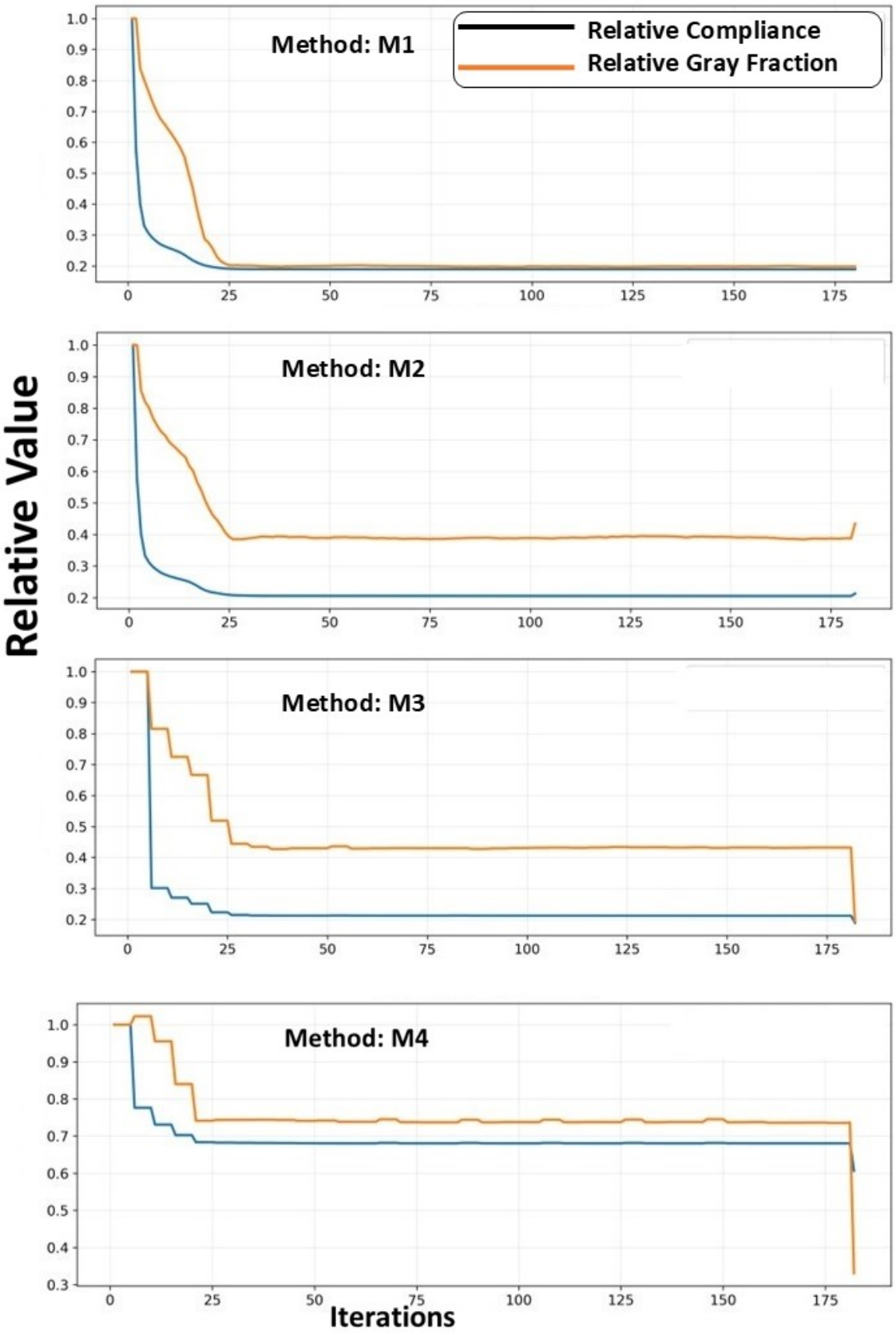

Figure 5.5: Relative compliance and relative gray fraction versus iteration for M1–M4 on the MID cantilever ($v_f = 0.50$).

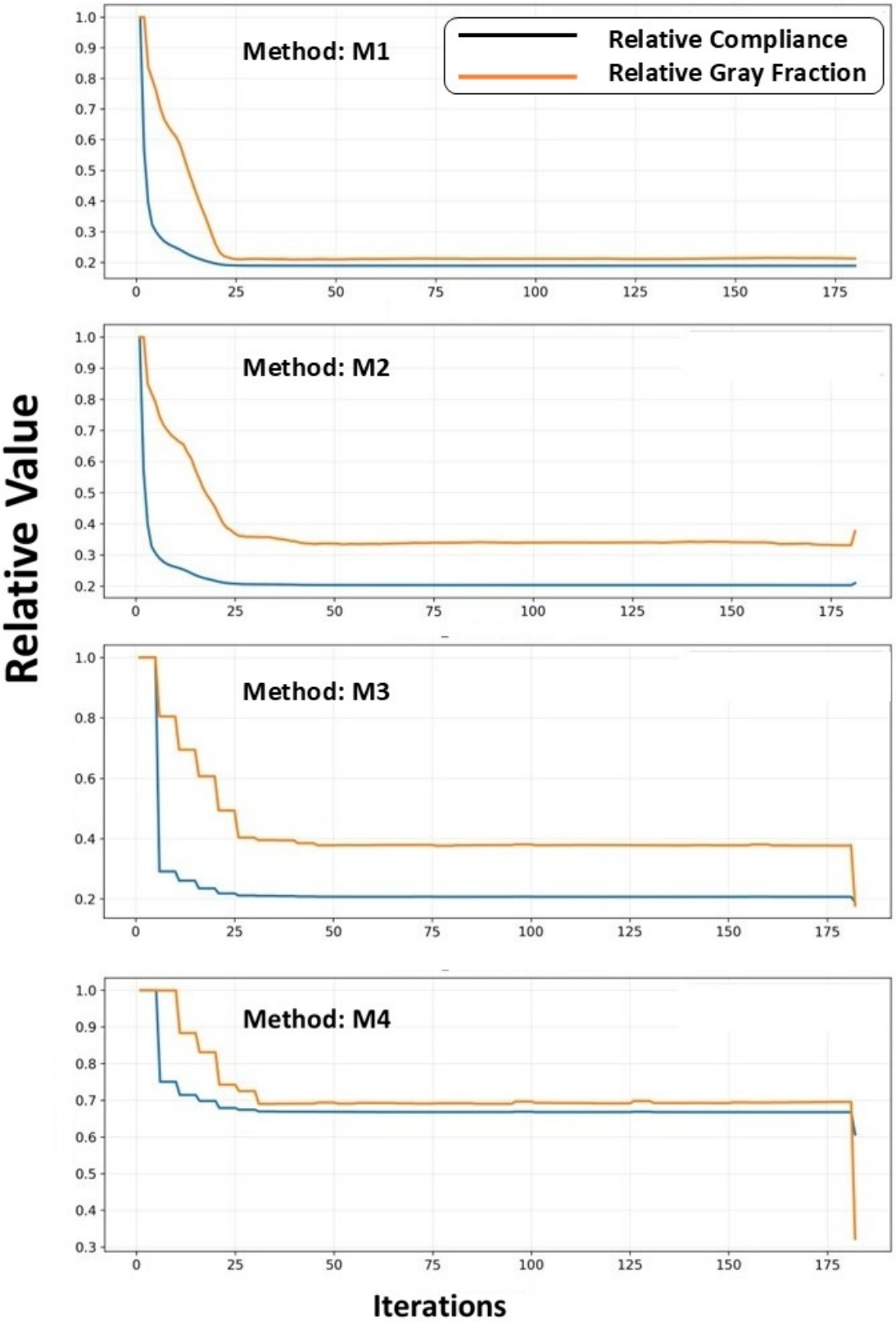

Figure 5.6: Relative compliance and relative gray fraction versus iteration for M1–M4 on the TIP cantilever ($v_f = 0.50$).

5.1.3 Determination of 2D compliance values

The fine-mesh compliance C is identical across all five seeds for every (problem, method) combination, yielding an intra-method standard deviation $\sigma_C = 0$. This outcome follows directly from a property of the OC update rule derived in Chapter 3: the ratio of the compliance sensitivity to the volume sensitivity that governs the density update is exactly invariant to the magnitude of the applied load, so the density trajectory is uniquely determined by the problem definition and mesh alone. Combined with the absence of any floating-point non-determinism in the sparse-direct linear solver, the full pipeline produces bitwise-identical compliance values across all seeds regardless of the random initialization used by the MFAS proposal mechanism.

Consequently, the $n = 5$ seeds in 2D serve exclusively to generate a distribution of *runtime* samples; compliance differences between methods are mathematical facts rather than statistical estimates. Paired Wilcoxon tests and Cohen's d_z on 2D compliance differences are inapplicable (a zero-variance paired sample violates the continuity assumption of the Wilcoxon test and renders $d_z = \bar{d}/\hat{s}_d$ undefined). All statistical tests in this section therefore apply to wall-clock time, from which speedup distributions are derived. The 3D experiments, where PCG convergence and load-offset seed effects introduce genuine variance, follow the full statistical protocol defined in Section 4.5.4.

5.1.4 Compliance and design quality

Table 5.1 summarizes the exact compliance, compliance gap Δ, gray fraction g, and volume fraction v_f for each method–problem pair.

Table 5.1: 2D main results: exact (deterministic) compliance C, compliance gap $\Delta = (C - C_{\mathrm{M1}})/C_{\mathrm{M1}} \times 100$ (%), gray fraction g, and mean ± standard deviation of wall-clock time $\bar{t} \pm \sigma_t$ (s) and speedup $\bar{S} \pm \sigma_S$ relative to M1 over $n = 5$ seeds. Empirical acceptance rate $\hat{\kappa}$ for M3/M4; "—" for M1/M2 (no acceptance step).

Problem	Method	C	Δ (%)	g	$\bar{t} \pm \sigma_t$ (s)	$\bar{S} \pm \sigma_S$	$\hat{\kappa}$
MBB	M1	244.809	0.000	0.224	55.69 ± 0.67	1.000 ± 0.000	—
	M2	272.237	+11.204	0.344	8.59 ± 0.15	6.487 ± 0.103	—
	M3	242.847	− 0.802	0.171	26.21 ± 0.41	2.125 ± 0.038	0.893
	M4	242.762	− 0.836	0.171	26.36 ± 0.45	2.113 ± 0.043	0.929
MID	M1	178.549	0.000	0.198	71.50 ± 0.57	1.000 ± 0.000	—
	M2	199.767	+11.884	0.434	10.56 ± 0.24	6.770 ± 0.110	—
	M3	178.355	− 0.108	0.194	31.09 ± 0.54	2.301 ± 0.032	0.917
	M4	178.212	− 0.188	0.194	31.48 ± 0.91	2.273 ± 0.067	0.861
TIP	M1	187.475	0.000	0.212	71.36 ± 1.13	1.000 ± 0.000	—
	M2	205.790	+ 9.769	0.376	10.93 ± 0.49	6.536 ± 0.191	—
	M3	187.598	+ 0.066	0.178	32.04 ± 1.34	2.229 ± 0.057	0.917
	M4	187.671	+ 0.104	0.183	32.17 ± 0.82	2.219 ± 0.047	0.944

Coarse-only optimization (M2) achieves large speedups (6.49–6.77×) but at the cost of severe compliance degradation ($\Delta = +9.77\,\%$ to $+11.88\,\%$). The gray fraction g nearly doubles relative to M1 (0.344–0.434 versus 0.198–0.224), reflecting the absence of projection cleanup on the coarse-prolongated density field. Method M2 is retained in the comparison as a lower-bound quality reference that quantifies the price of bypassing fine-mesh verification entirely; it is not considered a viable design method in practice. The MFAS methods M3 and M4 recover fine-mesh quality far more effectively than M2, but the sign and magnitude of Δ vary by problem geometry in ways that are physically informative and deserve careful attention.

On the MBB beam, both M3 and M4 yield *lower* compliance than M1 by 0.80 % and 0.84 % respectively. The MBB beam possesses a symmetric load path that the coarse mesh resolves with moderate fidelity. The MFAS coarse loop traverses a different region of the density space than the fine-mesh M1 trajectory; in these two cases this coincides with a lower final compliance at convergence, consistent with the coarse mesh providing broader exploratory coverage before cleanup. Whether this reflects a systematic advantage or an artifact of the specific initialization and convergence path is not resolved from the present experiments.

On the MID cantilever, M3 and M4 again produce lower compliance than M1 ($-0.11\,\%$ and $-0.19\,\%$), suggesting a similar exploration benefit at this geometry. On the TIP cantilever, however, M3 and M4 incur a *small positive* gap of $+0.07\,\%$ and $+0.10\,\%$ relative to M1. These values lie within the $\pm 1\,\%$ compliance range, which the 15-iteration fine-mesh cleanup phase is observed to correct (see the compliance trajectories in Figure 5.6). The magnitude is negligible from a structural standpoint. The main justification for MFAS on these problems is the $\approx 2\times$ runtime reduction, not a compliance improvement. All compliance values are exact to the precision stated. They do not carry statistical uncertainty.

Turning to the design quality metrics, M3 and M4 reduce g substantially relative to M2 across all problems: 50.2 %/50.4 % (MBB), 55.4 %/55.4 % (MID), and 52.8 %/51.4 % (TIP), consistent with the post-MFAS cleanup effectively eliminating intermediate-density material. Volume fraction errors remain below 0.006 % in all cases, verifying that the density pipeline $\boldsymbol{\rho} \to \tilde{\boldsymbol{\rho}} \to \bar{\boldsymbol{\rho}}$ enforces the prescribed constraint with high fidelity.

5.1.5 Runtime and speedup

The two-tailed paired Wilcoxon signed-rank test on runtime differences $d_i = t_{\mathrm{M1},i} - t_{k,i}$ ($n = 5$) yields $p = 0.0625$ for all six method–problem comparisons (M3 and M4 versus M1, three problems). With $n = 5$, the minimum achievable two-tailed p-value is $2 \times (1/32) = 0.0625$ (attained when all five signs are identical), so these results represent

the strongest evidence the sample size permits. Every M1 run is slower than every M3/M4 run on the same seed. The Cohen's d_z values range from 2.77 (MBB, M4) to 62.46 (TIP, M4), reflecting a practically large and consistent effect across all comparisons. The borderline $p = 0.0625$ is a consequence of the small n imposed by the experimental budget and should not be interpreted as evidence against the runtime benefit.

M3 achieves mean speedups of 2.13× (MBB), 2.30× (MID), and 2.23× (TIP). Runtime variance is low across all three problems ($\sigma_S \leq 0.06$). M4 achieves comparable wall-clock performance to M3: speedups of 2.11× (MBB), 2.27× (MID), and 2.22× (TIP). The post-inference density processing overhead of M4 does not produce a measurable wall-clock penalty relative to M3. Acceptance rates for M4 are 0.929 (MBB), 0.861 (MID), and 0.944 (TIP). The dominant per-iteration cost is fine-mesh FEA verification, not the coarse OC step, so acceptance-rate differences between M3 and M4 do not translate to a runtime advantage for either method.

5.1.6 MFAS diagnostics

The diagnostics reveal that M3 proposals span the full range of κ values across the optimization trajectory. Early iterations show low κ (coarse and fine compliances not yet aligned), transitioning to consistently high κ once the coarse topology enters the fine-mesh convergence basin. Rejected proposals cluster in the first quarter of iterations and become rare thereafter. M4 shows a different acceptance pattern: the U-Net warm-start produces a density field that is already near the fine-mesh attractor, resulting in high κ from the first few verifications on MBB.

5.2 Three-Dimensional Results

5.2.1 Problem setup and scope

Two 3D benchmarks are evaluated: a solid cantilever beam ($v_f = 0.10$; fine mesh $180 \times 60 \times 44 = 475{,}200$ H8 elements, coarse mesh $90 \times 30 \times 22 = 59{,}400$ elements) and an L-bracket with a passive void region ($v_f = 0.15$ over active elements; fine mesh $90 \times 90 \times 30 = 243{,}000$ H8 elements, coarse mesh $45 \times 45 \times 15 = 30{,}375$ elements). Each problem is solved with M1–M4 at $n = 3$ paired seeds. The 3D PCG linear solver exhibits genuine convergence variability across seeds due to load-offset perturbations and numerical tolerance interactions. Consequently, $\sigma_C > 0$ for the cantilever and statistical tests are applicable. Topology grids for the cantilever and L-bracket are shown in Figures 5.7 and 5.8. Baseline MFAS parameters for M3/M4 are $\epsilon_{\text{rel}} = 0.01$ and $f_{\text{verify}} = 5$ for both M3 and M4 on both 3D problems.

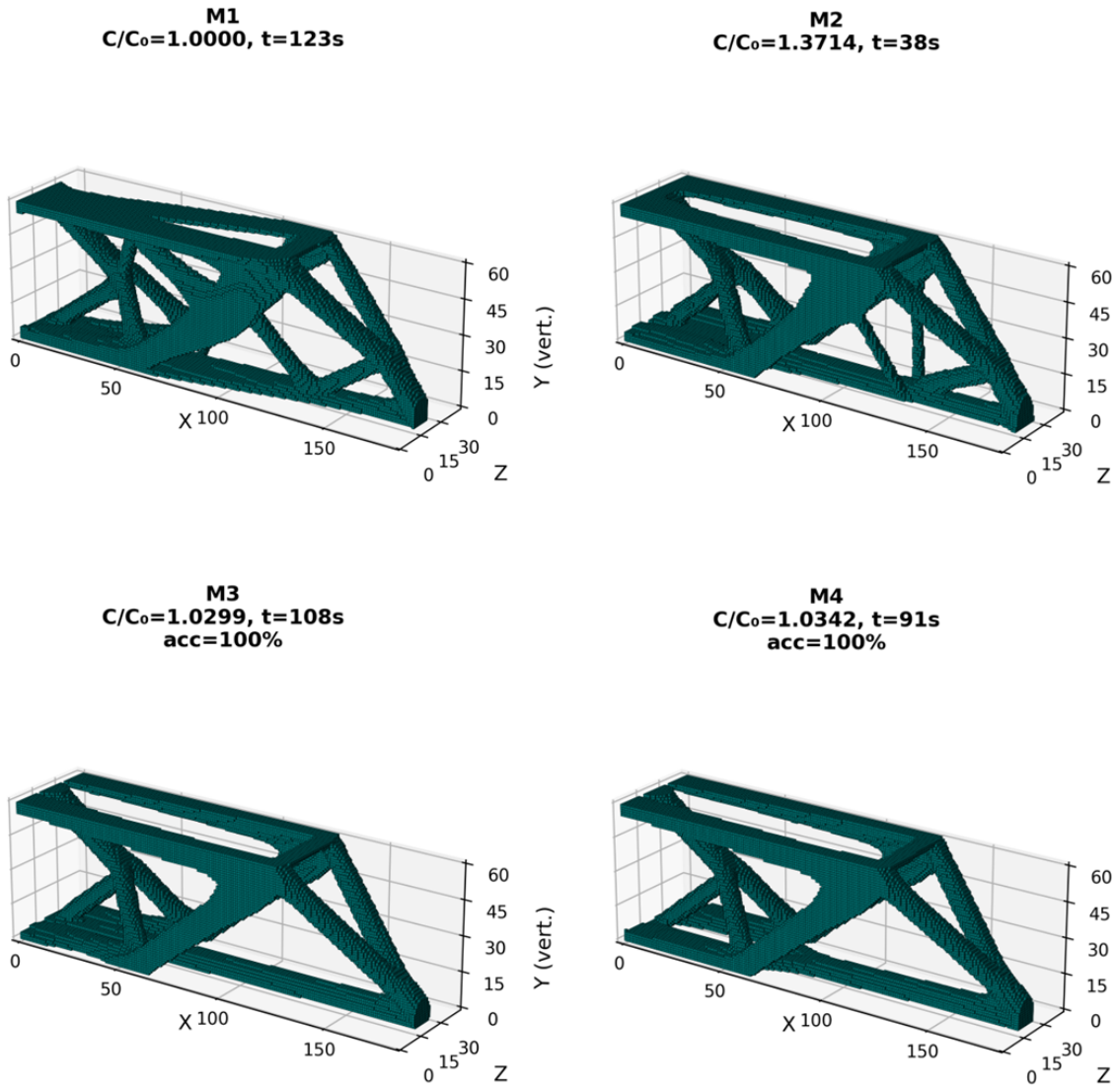

Figure 5.7: Optimized topologies for M1–M4 on the 3D cantilever ($v_f = 0.10$, median seed), with normalized compliance C/C_0 and wall-clock time t annotated per method.

5.2.2 Compliance and quality

With $n = 3$ seeds, the minimum achievable one-tailed Wilcoxon p-value is 0.125, so no result in this section crosses the conventional $\alpha = 0.05$ threshold; all compliance comparisons should be read as observed differences in the present sample rather than statistically resolved conclusions. Table 5.2 presents compliance and runtime statistics for all four methods across both 3D benchmark problems. For the L-bracket, $\sigma_C \approx 0$ for M2, M3, and M4 because the passive void mask deterministically constrains the active-domain density trajectory; CIs collapse to point estimates and Cohen's d_z is numerically large but carries no inferential content beyond establishing the sign of the mean difference. On the cantilever, M3 compliance exceeds M1 in two of three seeds (one-tailed $p = 0.875$; H_1: $C_{\mathrm{M3}} < C_{\mathrm{M1}}$ not supported), while M4 compliance exceeds M1 in all three seeds ($p = 1.00$). On the L-bracket, M3 and M4 both record

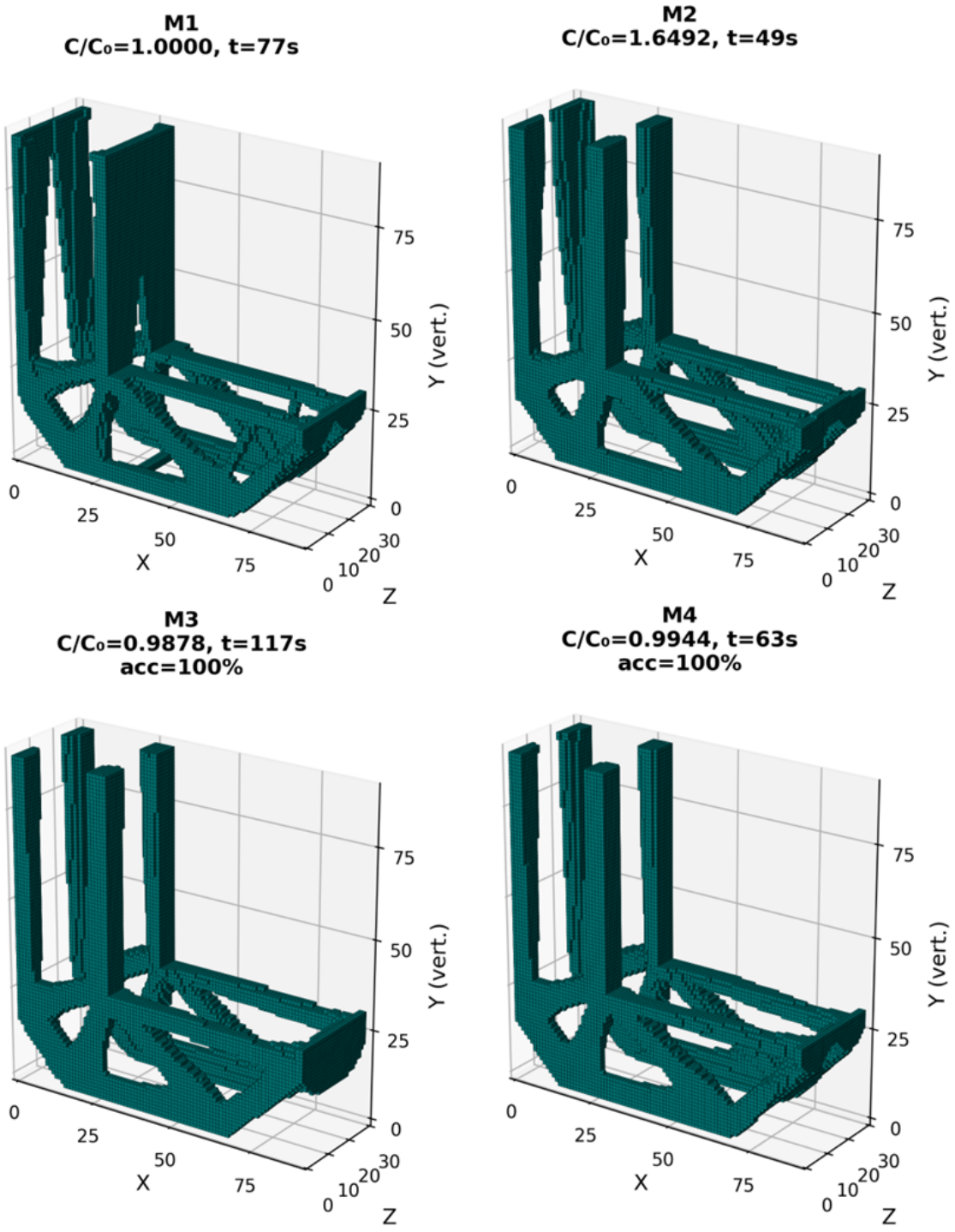

Figure 5.8: Optimized topologies for M1–M4 on the 3D L-bracket ($v_f = 0.15$ over active elements, median seed), with normalized compliance C/C_0 and wall-clock time t annotated per method.

Table 5.2: 3D main results: mean ± standard deviation of compliance $\bar{C} \pm \sigma_C$, wall-clock time t, normalized compliance C/C_{M1}, speedup $\bar{S}$, and empirical acceptance rate $\hat{\kappa}$ ($n = 3$ seeds per cell). 95 % Student-t confidence intervals on $\bar{C}$ for M3/M4; one-tailed Wilcoxon p-value and Cohen's d (pooled SD) versus M1. †Negative d denotes method compliance below M1 (lower = better); L-bracket M2 pooled-SD d exceeds 3000 and is reported as "large" to avoid implying precision not warranted at $n = 3$.

Problem	Method	$\bar{C} \pm \sigma_C$	95 % CI	C/C_{M1}	$\bar{t}$ (s)	$\bar{S}$	$\hat{\kappa}$	p / d
Cantilever	M1	12.087 ± 0.033	[12.006, 12.169]	1.000	117.9	1.00	—	—
	M2	15.962 ± 1.233	[12.898, 19.025]	1.321	40.2	2.95	—	$p = 1.00$; $d = 4.44$
	M3	12.331 ± 0.276	[11.646, 13.016]	1.020	79.7	1.64	0.833	$p = 0.875$; $d = 1.24$
	M4	12.509 ± 0.027	[12.443, 12.576]	1.035	92.8	1.28	1.00	$p = 1.00$; $d = 14.1$
L-bracket	M1	22.790 ± 0.007	[22.773, 22.806]	1.000	78.0	1.00	—	—
	M2	37.591 ± 0.000	—	1.650	48.7	1.60	—	$p = 1.00$; d large
	M3	22.516 ± 0.000	—	0.988	117.7	0.66	1.00	$p = 0.125$; $d = -58.6$ †
	M4	22.667 ± 0.000	—	0.995	62.6	1.25	1.00	$p = 0.125$; $d = -26.4$ †

compliance below M1 in all three seeds ($p = 0.125$)—the strongest evidence the sample size permits in that direction. For the M4 versus M3 paired comparison, $p = 0.625$ on the cantilever (no consistent direction) and $p = 1.00$ on the L-bracket (M3 consistently lower compliance than M4 in this sample). Figure 5.9 confirms that MFAS verification compliances plateau above M1's converged level on both problems, with the cleanup phase recovering only part of the remaining gap.

On the 3D cantilever, M2 records $\bar{C} = 15.96 \pm 1.23$ (+32.1 % above M1) with 2.95× speedup; the large σ_C reflects the sensitivity of the coarse-only trajectory to seed under load-offset perturbation. M3 reaches $\bar{C} = 12.331 \pm 0.276$ (+2.0 % above M1 in this sample, not resolved at $n = 3$) with a mean speedup of 1.64× and a mean acceptance rate of 0.833; the mean fine-mesh verification count is $\bar{n}_{\mathrm{fine}} = 35.7$ over $\bar{n}_{\mathrm{coarse}} = 98.3$ coarse iterations. M4 reaches $\bar{C} = 12.509 \pm 0.027$ (+3.5 % above M1; not resolved at $n = 3$) with a mean speedup of 1.28× and full acceptance ($\hat{\kappa} = 1.00$ in the present three-seed sample). The mean compliance difference between M3 and M4 (1.4 %) is not statistically resolved ($p = 0.625$).

The stage-wise timing decomposition explains the M3–M4 speed gap more precisely than aggregate wall-clock alone. For M4, total time comprises three stages: Stage 0 (U-Net inference, approximately 1.8 s, ≈2 % of total), Stage 1 (MFAS coarse loop with fine-mesh verification, mean 75.9 s), and Stage 2 (cleanup, mean 15.1 s). The coarse loop in Stage 1 is the dominant cost, and M4 runs a longer coarse loop than M3 ($\bar{n}_{\mathrm{coarse}} = 113.3$ versus 98.3 iterations) before the termination criterion is satisfied. The warm-start alters the coarse-loop starting point and trajectory, but in the present cantilever sample it does not shorten Stage 1 enough to compensate for the additional coarse iterations. Whether a different training corpus or a tighter termination criterion would change this balance is a question the present experiments cannot resolve.

Observed run-to-run compliance variability is notably different between M3 and M4:

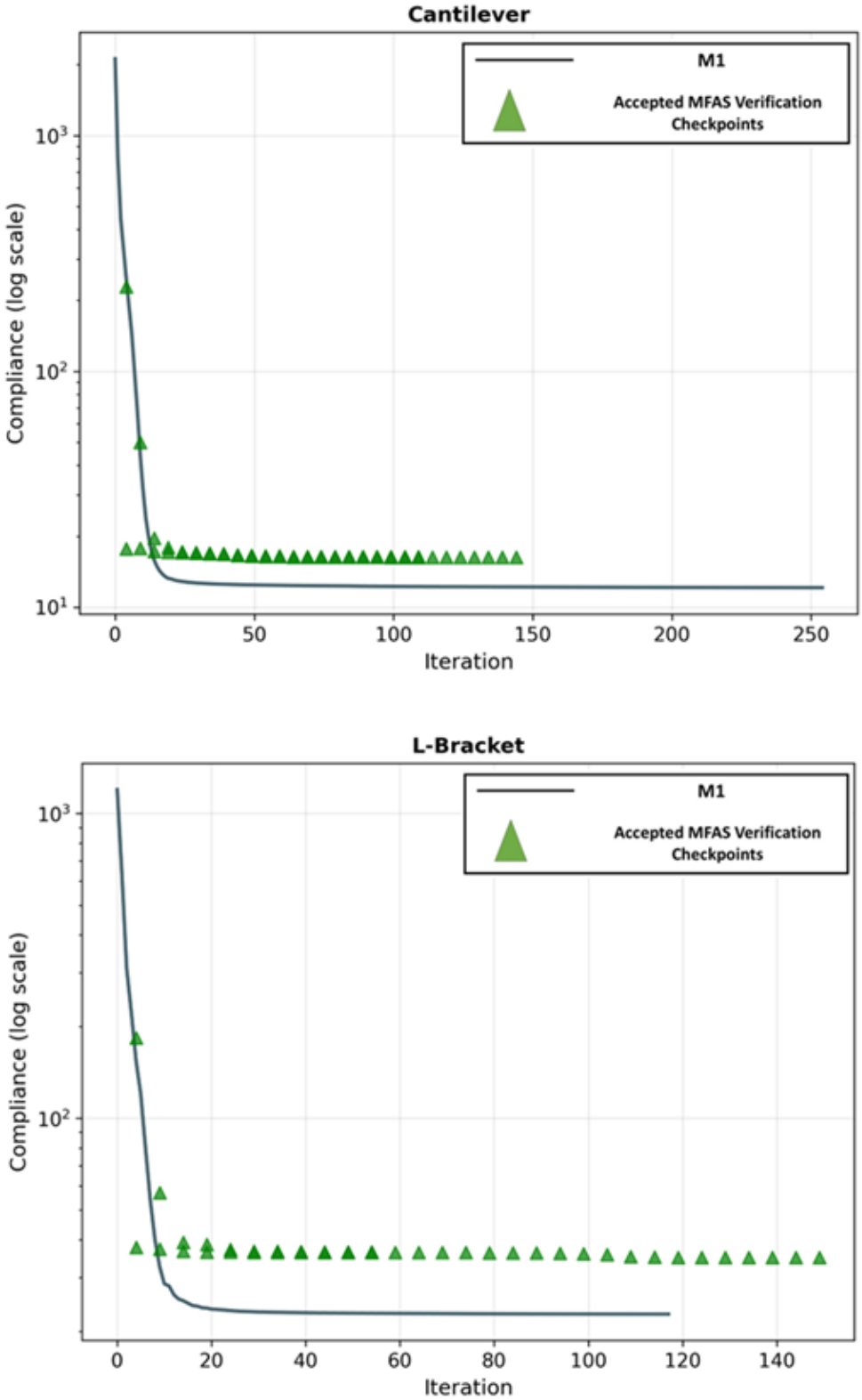

Figure 5.9: Compliance convergence history (log scale, median seed) for the 3D cantilever (left) and L-bracket (right). The M1 fine-mesh trajectory (solid line) continues descending below the plateau level of the accepted MFAS verification checkpoints (triangles), illustrating the compliance gap between the coarse-loop attractor and the fine-mesh optimum.

$\sigma_C = 0.276$ versus 0.027, a reduction of approximately 90 % in the three-seed sample. The warm-start appears to guide the coarse search toward a narrower region of the design space. Whether this lower variability generalizes beyond the present benchmark is an open question. Figure 5.10 shows that despite differing load-patch offsets, M4's best and worst seeds produce structurally near-identical topologies, visually confirming the low run-to-run compliance variability.

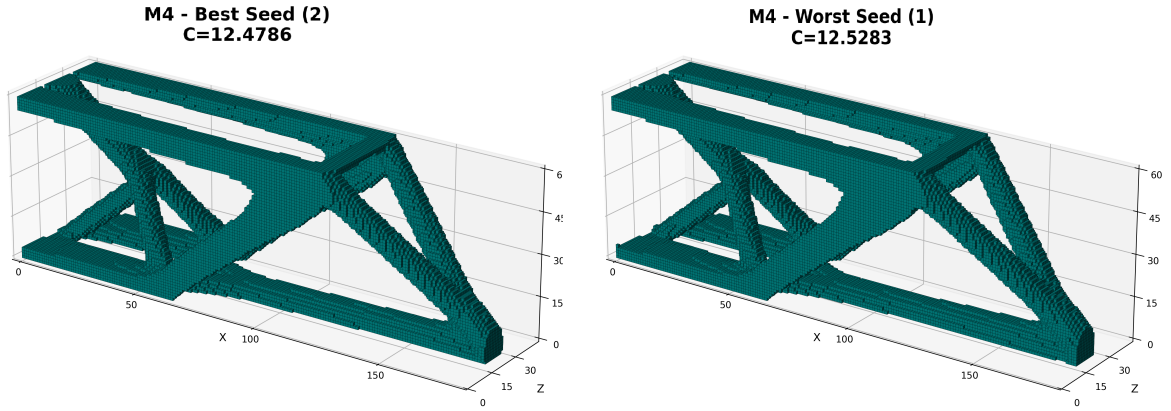

Figure 5.10: M4 best (seed 2, $C = 12.479$) and worst (seed 1, $C = 12.528$) topologies for the 3D cantilever ($\tau = 0.30$ iso-surface).

On the 3D L-bracket, M2 records a compliance penalty of +65.0 % relative to M1—substantially larger than on the cantilever—indicating that the passive void constraint amplifies the fidelity loss of coarse-only optimization under this mesh configuration. Under the present setup ($\epsilon_{\text{rel}} = 0.01$), M3 records $C = 22.516$ (−1.2 % relative to M1) and M4 records $C = 22.667$ (−0.5 % relative to M1) in the three-seed sample; both differences are not statistically resolved at $p = 0.125$. The near-zero σ_C for M3 and M4 on the L-bracket reflects the passive void mask pinning the active-domain trajectory to an essentially deterministic path. At the main-study $f_{\text{verify}} = 5$ setting, M3 requires 117.7 s ($\bar{S} = 0.66$, slower than M1) because fine-mesh verifications dominate runtime at this frequency on the stress-concentrated geometry; M4 achieves 62.6 s ($\bar{S} = 1.25$), where the Stage 0 U-Net cost (≈1.2 s) is small relative to the substantially shorter Stage 1 coarse loop (≈33.2 s mean versus ≈89.5 s for M3). The exploratory L-bracket ablation (Section 5.4.3) suggests that reducing f_{verify} to 3 can recover a 1.27× speedup for M3 at a 0.3 % compliance cost in the tested single-seed probe.

5.2.3 Runtime decomposition and acceptance behavior

As shown in Figure 5.11, the Stage 1 (MFAS coarse loop) dominates in all cases; Stage 0 U-Net inference contributes ≈2 % of total time. On the cantilever M4 runs a longer Stage 1 than M3, whereas on the L-bracket M4's Stage 1 is ≈60 % shorter, reversing the runtime relationship.

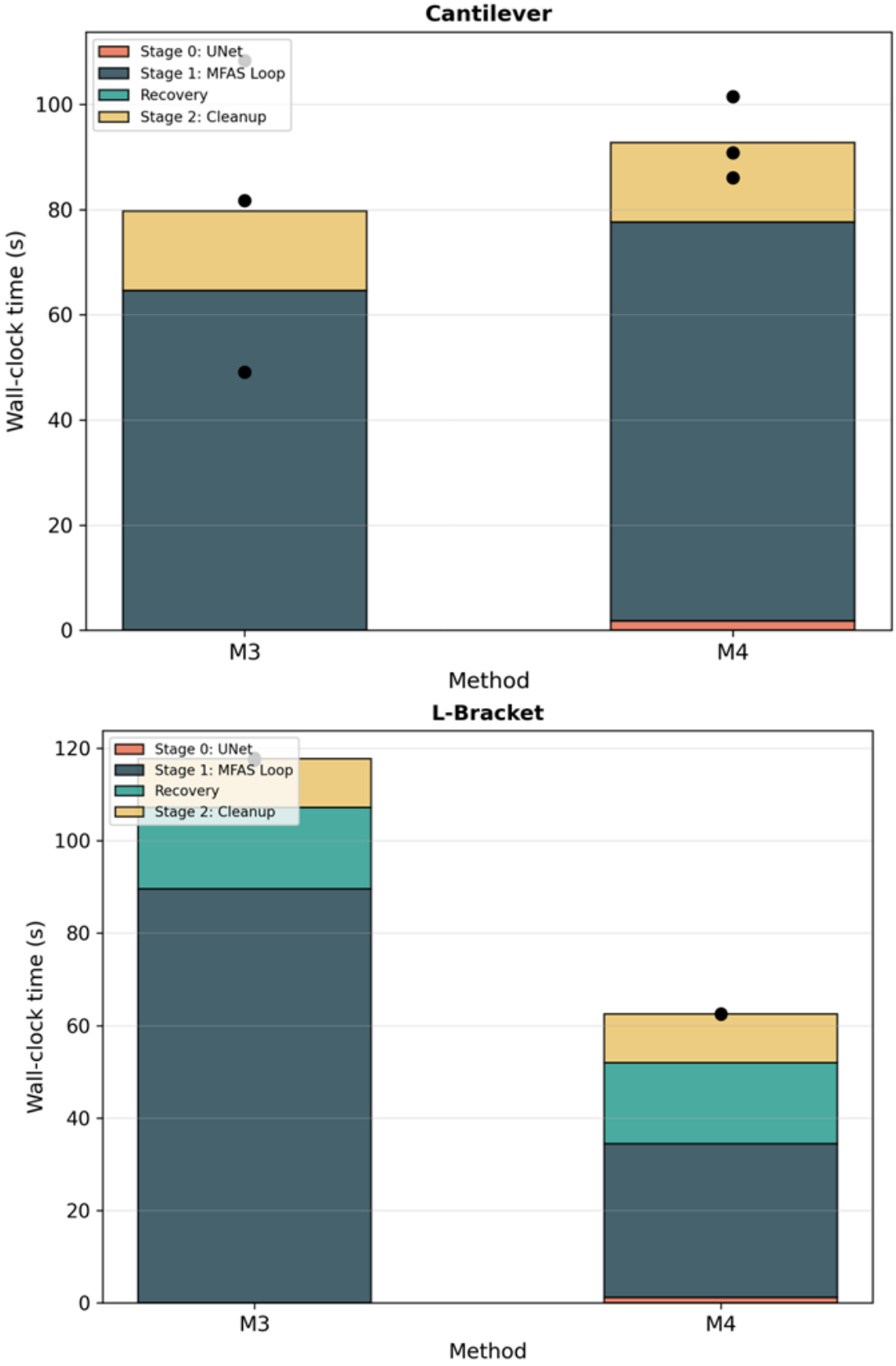

Figure 5.11: Stage-wise wall-clock decomposition for M3 and M4 on the 3D cantilever (left) and L-bracket (right).

The M3–M4 runtime difference is determined entirely by Stage 1 coarse-loop length. On the cantilever, M3 runtime variance is high ($\sigma_t = 29.7\,\text{s}$): one seed achieves $\hat{\kappa} = 0.50$ and completes in 49.1 s, while the other two reach $\hat{\kappa} = 1.00$ and take 81–108 s. This seed-dependent outcome is attributable to whether the coarse-loop trajectory enters the acceptance basin early or continues to generate rejected proposals. M4 shows lower runtime variance ($\sigma_t = 7.9\,\text{s}$), consistent with full acceptance and a more constrained coarse trajectory. On the L-bracket, both M3 and M4 are nearly deterministic ($\sigma_t \leq 0.3\,\text{s}$), matching the near-zero compliance variability.

The acceptance safeguard is present and active in the present experiments, but it should not be read as evidence of consistent selectivity from the main-table results alone. On the L-bracket and for M4 on the cantilever, acceptance is 100 % under the chosen defaults. The more informative evidence about how the safeguard responds to parameter changes comes from the ablation studies: in Ablation A, $f_{\text{verify}} = 10$ drops M3 acceptance to 0.20 while compliance remains close ($\bar{C} = 12.705$); in Ablation B, tightening ϵ_{rel} to 0.005 drops M4 acceptance to 0.55 and raises compliance by 0.6 %, while M3 acceptance drops to 0.75 with no compliance change. These ablation results, not the main-table acceptance rates, constitute the empirical evidence that the safeguard is exercise-capable and that its response is parameter-dependent.

5.3 Two-Dimensional Ablation Study

The 2D ablation experiments are conducted on the MBB beam using the paired seed subset $\{1, 2, 3\}$ ($n = 3$), with the same fine mesh and baseline parameters as the main study. The compliance values reported below are exact (deterministic), and the statistical interpretation follows Section 5.1.3.

5.3.1 Ablation A – Verification frequency (2D)

Method M3 is re-run with $f_{\text{verify}} \in \{3, 5, 10\}$ at fixed $\kappa_{\text{tr}} = 0.10$. The results are summarized in Table 5.3.

Table 5.3: 2D Ablation A: effect of f_{verify} on M3 (MBB beam, $n = 3$ seeds). C is exact (deterministic). Runtime statistics are means over three seeds; speedup relative to $\bar{t}_{\text{M1}} = 55.69\,\text{s}$.

f_{verify}	C	$\bar{t} \pm \sigma_t$ (s)	$\bar{S}$	$\hat{\kappa}$
3	242.850	33.15 ± 0.16	1.68	0.935
5	242.847	28.69 ± 0.22	1.94	0.893
10	242.718	22.37 ± 0.40	2.49	0.857

Increasing f_{verify} from 3 to 10 reduces mean wall-clock time by 32 % (from 33.2 s to

22.4 s) and lowers the acceptance rate by 0.078 absolute. The compliance change is negligible: $|C(f = 10) - C(f = 5)| = 0.129$, a 0.05 % difference that lies well within the cleanup-stage correction capacity. The time profile is monotonically decreasing with f_{verify}. More coarse iterations between verifications means fewer fine-mesh FEA calls per unit optimization progress. The 2D coarse loop does not exhibit trajectory drift at $f_{\text{verify}} = 10$ because the problem geometry lacks the stress concentrations and passive constraints that destabilize the 3D L-bracket coarse trajectory.

The $f_{\text{verify}} = 5$ setting provides the best compliance–runtime balance: 1.94× speedup with $\hat{\kappa} = 0.893$ and $C = 242.847$. The $f_{\text{verify}} = 3$ setting increases acceptance marginally (+0.042) but reduces speedup to 1.68×, eroding most of the MFAS benefit for negligible compliance gain.

5.3.2 Ablation B – Acceptance threshold (2D)

Both M3 and M4 are re-run with $\kappa_{\text{tr}} \in \{0.05, 0.10, 0.20\}$ at fixed $f_{\text{verify}} = 5$. Table 5.4 presents the results.

Table 5.4: 2D Ablation B: effect of κ_{tr} on M3 and M4 (MBB beam, $n = 3$ seeds). C and $\hat{\kappa}$ are deterministic. Runtime means over three seeds.

κ_{tr}	Method	C	$\bar{t}$ (s)	$\hat{\kappa}$
0.05	M3	242.847	28.10	0.893
0.10	M3	242.847	26.88	0.893
0.20	M3	242.847	27.13	0.893
0.05	M4	242.761	27.33	0.964
0.10	M4	242.762	27.46	0.929
0.20	M4	242.762	26.90	0.929

M3 is completely insensitive to κ_{tr} over the tested range: compliance, runtime, and acceptance rate are identical at all three threshold values. This is consistent with the theoretical expectation: with uniform initialization, the trust ratio κ_j for individual M3 proposals is either well above or well below κ_{tr}, so the threshold value in $[0.05, 0.20]$ never constitutes an active constraint.

M4 shows mild sensitivity. At $\kappa_{\text{tr}} \geq 0.10$, the acceptance rate drops from 0.964 to 0.929, indicating that a subset of U-Net-warm-started proposals have $0.05 \leq \kappa_j < 0.10$. Tightening the threshold rejects these marginal proposals, forcing additional coarse iterations. Compliance is essentially unchanged across all three settings ($C = 242.761$ at $\kappa_{\text{tr}} = 0.05$; $C = 242.762$ at $\kappa_{\text{tr}} = 0.10$ and 0.20), consistent with the rejected proposals carrying negligible structural information.

The ablation evidence supports $f_{\text{verify}} = 5$ and $\kappa_{\text{tr}} = 0.10$ as the recommended 2D defaults. Under this configuration, M3 achieves 1.94× speedup and $\hat{\kappa} = 0.893$ with a compliance within 0.80 % of M1 on the most sensitive geometry tested. For

maximum throughput at slightly reduced acceptance, $f_{\text{verify}} = 10$ is viable (2.49× speedup; compliance 0.05 % below the $f = 5$ setting). The κ_{tr} parameter is inert for M3 and can be left at its default; for M4, any value in $[0.05, 0.20]$ yields negligible compliance change, and $\kappa_{\text{tr}} = 0.10$ is a reasonable default.

5.4 Three-Dimensional Ablation Study

The 3D ablation experiments evaluate the same parameters in the 3D setting, where PCG solver dominance makes each fine-mesh verification significantly more expensive than in 2D. All 3D ablations use the cantilever as the primary case (Ablations A and B); Ablation A2 applies the frequency sweep to the L-bracket.

5.4.1 Ablation A – Verification frequency (3D cantilever)

M3 is re-run on the 3D cantilever with $f_{\text{verify}} \in \{3, 5, 10\}$ at $\epsilon_{\text{rel}} = 0.01$ ($n = 3$ seeds per setting). Table 5.5 summarizes the results. Figure 5.12 reveals that the acceptance collapse at $f_{\text{verify}} = 10$ is uniform across all seeds, and that the high runtime variance at $f_{\text{verify}} = 5$ is driven by seed 1's partial acceptance ($\hat{\kappa} = 0.50$).

Table 5.5: 3D Ablation A: effect of f_{verify} on M3 (cantilever, $n = 3$ seeds, $\epsilon_{\text{rel}} = 0.01$). Speedup relative to $\bar{t}_{\text{M1}} = 117.9\,\text{s}$.

f_{verify}	$\bar{C} \pm \sigma_C$	$\bar{t}$ (s)	$\bar{S}$	$\hat{\kappa}$
3	12.776 ± 0.296	74.6	1.58	0.833
5	12.331 ± 0.276	79.7	1.48	0.833
10	12.705 ± 0.206	70.7	1.67	0.200

No single f_{verify} setting dominates across all metrics. $f_{\text{verify}} = 10$ gives the shortest mean wall-clock time (70.7 s, 1.67× speedup) but acceptance collapses to $\hat{\kappa} = 0.20$, indicating that the coarse trajectory drifts between verifications. $f_{\text{verify}} = 5$, the main-study setting, yields the lowest mean compliance ($\bar{C} = 12.331 \pm 0.276$) with moderate acceptance ($\hat{\kappa} = 0.833$) and a 1.48× speedup, representing the best quality–runtime balance among the three settings. $f_{\text{verify}} = 3$ gives a slightly higher mean compliance ($\bar{C} = 12.776 \pm 0.296$) and an intermediate runtime (74.6 s, 1.58×) with the same acceptance rate (0.833); despite the more frequent verification cadence, the final compliance remains slightly above the $f_{\text{verify}} = 5$ case, suggesting that additional verification overhead does not improve the accepted coarse trajectory enough to compensate for the reduced number of productive coarse OC steps between checks. Runtime variance is high at $f_{\text{verify}} \in \{3, 5\}$ ($\sigma_t > 29\,\text{s}$) because one seed achieves $\hat{\kappa} = 0.50$ and finishes in approximately half the time of the other two seeds; at $f_{\text{verify}} = 10$, all seeds accept at the same low rate and runtimes are consistent ($\sigma_t = 2.3\,\text{s}$). The variance–acceptance

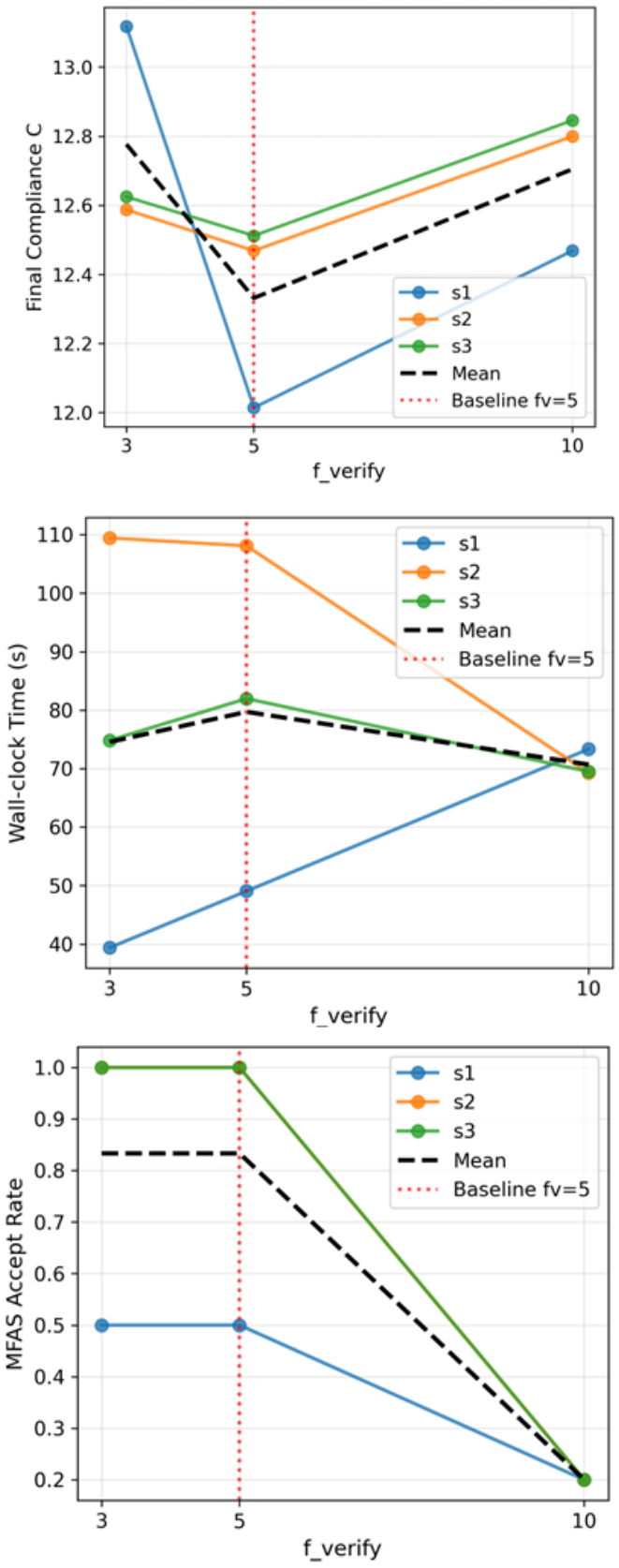

Figure 5.12: 3D Ablation A: per-seed and mean compliance (left), wall-clock time (centre), and MFAS acceptance rate $\hat{\kappa}$ (right) for M3 on the 3D cantilever across $f_{\text{verify}} \in \{3, 5, 10\}$. Seed 1 (blue) achieves $\hat{\kappa} = 0.50$ at $f_{\text{verify}} = 5$; acceptance collapses to $\hat{\kappa} = 0.20$ for all seeds at $f_{\text{verify}} = 10$, indicating systematic coarse-trajectory drift.

trade-off does not arise in 2D because the OC solver is deterministic.

5.4.2 Ablation B – Relative acceptance tolerance (3D cantilever)

Both M3 and M4 are re-run with $\epsilon_{\text{rel}} \in \{0.005, 0.01, 0.02, 0.05, 0.10\}$ at $f_{\text{verify}} = 5$ for both methods ($n = 2$ seeds per cell). Results are presented in Table 5.6.

Table 5.6: 3D Ablation B: effect of ϵ_{rel} on M3 and M4 (cantilever, $n = 2$ seeds per cell). $C_{\text{trial}} \leq (1 + \epsilon_{\text{rel}})\, C_{\text{best}}$ is the acceptance criterion.

ϵ_{rel}	Method	$\bar{C}$	$\hat{\kappa}$
0.005	M3	12.241	0.75
0.010	M3	12.241	0.75
0.020	M3	12.241	0.75
0.050	M3	12.241	1.00
0.100	M3	12.241	1.00
0.005	M4	12.578	0.55
0.010	M4	12.503	1.00
0.020	M4	12.503	1.00
0.050	M4	12.503	1.00
0.100	M4	12.503	1.00

M3 is insensitive to ϵ_{rel} in final compliance across the full tested range ($\bar{C} = 12.241$ for all five settings). Acceptance rate transitions from 0.75 at tight tolerances (≤ 0.02) to 1.00 at looser tolerances (≥ 0.05): tighter criteria require more verification passes before termination, but do not alter the final topology in the present sample.

M4 is sensitive only at the very tightest tolerance: at $\epsilon_{\text{rel}} = 0.005$, acceptance drops to 0.55 and mean compliance rises to $\bar{C} = 12.578$. For $\epsilon_{\text{rel}} \geq 0.01$, acceptance is 1.00 and compliance stabilizes at $\bar{C} = 12.503$. The main experiment uses $\epsilon_{\text{rel}} = 0.01$, which is the tightest setting compatible with full M4 acceptance in the present two-seed probe; the compliance difference between $\epsilon_{\text{rel}} = 0.005$ and $\epsilon_{\text{rel}} \geq 0.01$ is 0.075 (+0.6 %), indicating that the U-Net warm-start prediction falls just outside the 0.5 % acceptance band in one of two seeds.

Taken together, the 3D ablations provide the clearest message about parameter sensitivity in this study. Ablation A shows that $f_{\text{verify}} = 5$ is the best-balanced setting in the tested cantilever grid: $f_{\text{verify}} = 3$ is slower and gives higher mean compliance, while $f_{\text{verify}} = 10$ is faster but acceptance collapses to 0.20, indicating coarse-trajectory drift that the fine-mesh verification is no longer correcting efficiently. Ablation B shows that varying ϵ_{rel} primarily changes acceptance behaviour, with negligible effect on final compliance over the range $[0.01, 0.10]$; the only compliance change observed is at $\epsilon_{\text{rel}} = 0.005$ for M4, where the tighter gate and reduced acceptance coincide with a small compliance increase. This is the cleanest evidence in the chapter that the two

hyperparameters control distinct aspects of the quality–speed trade-off: f_{verify} governs how often the fine-mesh acts as a filter on coarse proposals, while ϵ_{rel} governs how strictly individual proposals must improve on the current best.

5.4.3 Ablation A2 – Verification frequency (3D L-bracket)

M3 is re-run on the 3D L-bracket with $f_{\text{verify}} \in \{3, 5, 10\}$ (single seed; $n = 1$). The main-study result uses $f_{\text{verify}} = 5$, which gives $C = 22.516$, $t = 117.9\,\text{s}$, $\hat{\kappa} = 1.00$, and $\bar{S} = 0.66$—slower than M1. At $f_{\text{verify}} = 3$: $C = 22.587$ (+0.3 % relative to $f = 5$), $t = 61.5\,\text{s}$, $\hat{\kappa} = 1.00$, $\bar{S} = 1.27$. At $f_{\text{verify}} = 10$: $C = 22.645$ (+0.6 % relative to $f = 5$), $t = 81.3\,\text{s}$, $\hat{\kappa} = 0.20$, $\bar{S} = 0.96$.

$f_{\text{verify}} = 3$ provides the best overall balance for the L-bracket in this single-seed exploratory probe: it achieves a speedup above 1× with high acceptance and a compliance only 0.3 % above the main-study result. $f_{\text{verify}} = 10$ degrades acceptance to 0.20, consistent with the cantilever ablation, where the same setting also gave $\hat{\kappa} = 0.20$. The low acceptance at $f_{\text{verify}} = 10$ reflects the stress concentration at the L-bracket notch: the coarse mesh accumulates density in the passive region over ten iterations between corrections, which fine-mesh verifications consistently reject. This finding is consistent with the cantilever ablation results, but because it is based on $n = 1$, it should be treated as indicative rather than definitive; a multi-seed study would be required to confirm the recommendation for geometrically complex problems with passive constraints.

5.5 Cross-Case Synthesis and Practical Guidance

5.5.1 When MFAS provides the greatest benefit

In the present experiments, MFAS (M3) delivers a consistent wall-clock benefit when two conditions hold simultaneously: fine-mesh FEA is the dominant cost, making coarse iterations cheap relative to fine-mesh verification solves; and the coarse mesh provides adequate fidelity for proposals to be accepted at a reasonable rate. The 2D results are consistent with both conditions: 2.13–2.30× speedup is observed across all three benchmark geometries. In 3D, the benefit is geometry-dependent. On the cantilever, M3 achieves a mean speedup of 1.64× with a compliance +2.0 % above M1 in the tested sample. On the L-bracket at the main-study $f_{\text{verify}} = 5$ setting, M3 is slower than M1 ($\bar{S} = 0.66$) because the high frequency of fine-mesh verifications needed to track the notch stress concentration outweighs the coarse-loop savings; the exploratory single-seed ablation shows that $f_{\text{verify}} = 3$ is associated with $\bar{S} = 1.27$ at a modest compliance cost, though this finding requires multi-seed validation. M4 shows a more consistent 3D runtime profile: 1.28× on the cantilever and 1.25× on the L-bracket in the present sample.

M2 achieves greater raw speedup (1.60×–6.77× across 2D and 3D problems) but at compliance penalties of +10 % to +65 % relative to M1; these penalties are not acceptable in structural optimization practice and M2 serves only as a lower-bound quality reference.

5.5.2 Role of the U-Net warm-start across problem scales

The U-Net warm-start (M4) has a qualitatively different effect in 2D and 3D. In 2D, where the OC solver is fully deterministic, compliance values are identical within each method across seeds. The warm-start does not alter this deterministic outcome. M3 and M4 produce comparable wall-clock times in 2D (2.11–2.30× and 2.11–2.27× respectively), and neither method consistently dominates the other. In 3D, where the PCG solver introduces genuine seed-to-seed variability, the picture is different. On the cantilever, M4 shows $\sigma_C = 0.027$ across three seeds, compared to $\sigma_C = 0.276$ for M3—a reduction in observed run-to-run compliance variability of approximately 90 % in the present sample. This suggests the warm-start guides the coarse search toward a narrower region of the design space. Whether this lower variability generalises beyond the present benchmark and seed count is an open question.

The stage-wise decomposition is the most instructive framing for M4 versus M3 at the system level. M4 changes the optimization trajectory by providing a non-random starting density field, but its total wall-clock benefit depends on whether the Stage 1 coarse-loop savings offset Stage 0 overhead and any longer plateau behaviour before the termination criterion is satisfied. On the present cantilever, M4's Stage 1 loop is longer (113 versus 98 mean coarse iterations), so M4 is slower in aggregate (1.28× versus 1.64× for M3). On the L-bracket, M4's Stage 1 is dramatically shorter (33.2 s versus 89.5 s for M3), producing a net speedup advantage. The practical guidance is therefore problem-dependent: M3 is observed to maximise throughput on the cantilever in this sample, while M4 is observed to provide both faster runtime and lower compliance variability on the L-bracket.

5.6 Limitations

The present study adopts a fixed 4 : 1 element-count coarse-to-fine reduction ratio for all experiments, and the achieved speedups, acceptance rates, and compliance gaps are therefore specific to this pairing. A smaller ratio (e.g., 2 : 1) would reduce the coarse-mesh computational advantage; a larger ratio (e.g., 8 : 1) would increase coarse-loop efficiency but degrade prolongation fidelity and likely reduce $\hat{\kappa}$, as observed when $f_{\text{verify}} = 10$ is combined with the geometrically complex L-bracket (Section 5.4.3). Exploration of alternative mesh pairings, including problem-adaptive ratios, is deferred

to future work.

A second limitation is that the MFAS acceptance criterion $C_{\text{trial}} \leq (1 + \epsilon_{\text{rel}})\, C_{\text{best}}$ is formulated exclusively for a single compliance objective. Extension to multi-objective, stress-constrained, or manufacturing-constrained formulations requires modification of the acceptance logic and the density pipeline $\boldsymbol{\rho} \to \tilde{\boldsymbol{\rho}} \to \bar{\boldsymbol{\rho}}$. Whether the acceptance rates and speedups observed here persist under richer objective functions is an open question that the present experiments cannot resolve.

The benchmark suite is also limited in the variety of load cases and structural archetypes it encompasses. The three 2D problems share the archetype of simply-supported or cantilevered beams under concentrated or distributed loads, and while the 3D L-bracket introduces geometric complexity through its passive void constraint, it remains a single-objective static problem. Multi-load, thermally-coupled, or dynamically-loaded configurations may exhibit qualitatively different coarse-to-fine fidelity gaps, potentially destabilizing the MFAS acceptance logic in ways not captured here. The small positive compliance gap observed on the TIP cantilever (+0.07 % to +0.10 %) is an early indicator that certain load configurations can resist the MFAS coarse-loop exploration benefit, and a broader survey of load diversity would be needed to characterize this sensitivity more fully.

A related limitation concerns the generalization scope of the U-Net warm-start. Separate 3D U-Net models were trained for each benchmark problem using problem-specific 3D training corpora. The present study therefore evaluates the warm-start under a proof-of-concept regime: the training sets are relatively small and restricted to a single geometry and load family per problem. Whether the observed variance-reduction benefit—σ_C reduced from 0.276 to 0.027 on the cantilever, approximately 90 %—persists under broader geometry variation, different load configurations, or out-of-distribution meshes remains an open question. Quantifying this generalization scope, and extending the training corpus to span multiple structural archetypes, is an identifiable direction for future work.

The 3D wall-clock speedups vary considerably across methods and problems. M3 ranges from 0.66× (L-bracket at $f_{\text{verify}} = 5$, slower than M1) to 1.64× (cantilever); M4 gives 1.25–1.28× across both problems. These values are lower than their 2D counterparts (2.11–2.30×), and the difference is substantially attributable to the PCG linear solver cost for 3D fine-mesh FEA. On the hardware employed, the PCG solver constitutes a large fraction of total iteration time, so the relative benefit of replacing fine-mesh solves with coarse-mesh iterations is proportionally smaller than in 2D, where a sparse-direct factorization is used. The speedups are therefore hardware- and solver-specific. A more efficient fine-mesh solver—such as a multigrid-preconditioned PCG or a GPU-accelerated direct factorization—would alter the coarse-to-fine cost ratio and could yield higher MFAS speedups without any change to the algorithmic design.

Finally, the 3D experiments are constrained by limited statistical power. With only $n = 3$ seeds per configuration, the minimum achievable one-tailed Wilcoxon p-value is 0.125, which falls above the conventional $\alpha = 0.05$ threshold and precludes formal rejection of the null hypothesis. On the cantilever, MFAS compliance exceeds M1 for most seeds (M3: $p = 0.875$; M4: $p = 1.00$), while on the L-bracket both M3 and M4 record compliance below M1 in all three seeds ($p = 0.125$)—the strongest evidence the sample size permits in that direction. These results should be interpreted with appropriate caution given the small n. A larger seed ensemble of $n \geq 10$ would provide adequate power for formal hypothesis testing in 3D, but generating such an ensemble was infeasible within the available computational budget.

Chapter 6

Conclusion and Future Work

6.1 Summary of Contributions

This dissertation addressed a central problem in density-based topology optimization: coarse-mesh or surrogate-based acceleration is only practically useful if the final fine-mesh solution quality remains trustworthy and quantifiable. Achieving speed at the cost of uncontrolled solution degradation is not a credible trade-off in engineering practice. The framework developed here treats runtime quality control as the central design objective, with acceleration as a consequence rather than a premise.

The primary contribution is a *verification-guided multifidelity framework* for compliance-based topology optimization. Coarse-mesh iterations serve as low-cost proposal steps. Periodic fine-mesh evaluations act as acceptance tests. The central novelty lies not in coarse acceleration alone, but in the explicit accept/reject logic that keeps the coarse search anchored to fine-mesh quality at every verification step. The 2D and 3D formulations differ in their acceptance criterion design—a trust ratio κ in 2D and a relative compliance tolerance ϵ_{rel} in 3D—but both share a rollback-and-recovery mechanism that restores the last accepted fine state whenever a proposal is rejected. This prevents silent drift along unpromising coarse trajectories.

The methodological contribution is the explicit design and implementation of the runtime safeguard itself. Several coarse or surrogate acceleration methods in the topology optimization literature incorporate implicit quality controls, but explicit runtime safeguards—ones that evaluate every coarse proposal against a fine-mesh criterion before acceptance—are rarely treated as the primary algorithmic contribution. The present framework treats the safeguard as such: every proposal is evaluated against a fine-mesh verification criterion, and rejected proposals do not propagate into the accepted design. The resulting diagnostic—the acceptance rate $\hat{\kappa}$—is an interpretable empirical indicator of coarse-model alignment under the current verification policy. This is compatible with both classical uniform initialization (M3) and learned warm-starting

via a trained U-Net (M4).

The empirical contribution is a systematic study of the speed–quality–reliability trade-off across three 2D and two 3D benchmark problems, spanning 142 total runs. The study documents compliance retention, wall-clock reduction, acceptance behavior, and sensitivity to verification frequency and acceptance tolerance. The ablation studies—which vary f_{verify} and κ_{tr} (2D) or ϵ_{rel} (3D)—provide the clearest evidence of how the safeguard responds to parameter changes. Together, the main experiments and ablations establish empirically that the framework achieves meaningful speedup while bounding compliance loss within an interpretable and adjustable range.

The ML integration contribution is framed carefully. The thesis does not treat the U-Net as a replacement for physics-based optimization. It treats it as a proposal generator embedded within the safeguarded loop. The verification mechanism applies identically whether the starting field comes from a random initialization or a learned prediction. This means the reliability of the framework does not depend on prediction quality. A poor U-Net prediction is rejected and the optimizer reverts to the last accepted state. The contribution of M4 is therefore not a claim of unconditional ML superiority, but a demonstration that learned warm-starting can be integrated safely into a verified optimization loop without relaxing the quality assurance of the framework.

6.2 Answers to Research Questions

Chapter 1 formulated four research questions. Each is answered below using the experimental evidence from Chapters 4 and 5.

RQ1 — Acceleration

Can a runtime-verified multifidelity strategy reduce the computational cost relative to a fine-mesh SIMP baseline while maintaining a comparable final compliance?

The experimental evidence suggests an affirmative answer, though with method- and problem-specific qualifications that deserve careful consideration. In the two-dimensional setting, M3 achieves wall-clock speedups in the range 2.13–2.30× relative to the fine-mesh baseline M1, across all three benchmark problems and all five repeated seeds. The associated compliance gaps remain within ±0.80 % of the M1 reference (Table 5.1), suggesting that the runtime safeguard is effective at limiting compliance degradation. Notably, these results are obtained with a fixed parameter configuration across the MBB, MID, and TIP benchmarks, without problem-specific re-tuning.

In 3D, the speedup is geometry-dependent. On the cantilever, M3 achieves a mean speedup of 1.64× with a compliance +2.0 % above M1 in the three-seed sample. On the L-bracket at the main-study setting ($f_{\text{verify}} = 5$), M3 is slower than M1 ($\bar{S} = 0.66$)

because the dense verification schedule required to track the notch stress concentration outweighs the coarse-loop savings. M4 achieves 1.28× on the cantilever and 1.25× on the L-bracket. On the cantilever, M3 offers the stronger mean speed–quality compromise; on the L-bracket, M4 is more favorable.

Acceleration alone is not sufficient as a criterion. M2, the coarse-only method without verification, achieves 6.5–6.8× speedup in 2D but at compliance penalties of +9.8 % to +11.9 % and dramatically elevated gray fractions. In 3D, M2 penalties reach +32.1 % on the cantilever and +65.0 % on the L-bracket. These are not acceptable outcomes in structural practice. The practical contribution of the framework is therefore not speedup per se, but speedup with bounded and quantifiable compliance cost.

RQ2 — Reliability

Does the acceptance mechanism prevent the accumulation of coarse-model errors and avoid the silent solution degradation characteristic of unverified coarse or surrogate approaches?

The evidence supports an affirmative answer. The contrast with M2 is the central argument. Unverified coarse-only optimization (M2) degrades compliance substantially in every tested problem, both in 2D and in 3D. M3 and M4, which share the same acceptance and rollback logic, produce final compliance values close to the fine-mesh baseline in all cases. No M3 or M4 run produced a final compliance at or near the M2 degradation level. The rollback mechanism correctly intercepts quality-degrading proposals and restores the last verified state.

The main-table acceptance rates should not be overinterpreted on their own. At the default settings, both M3 and M4 achieve full acceptance ($\hat{\kappa} = 1.00$) on the L-bracket and M4 achieves $\hat{\kappa} = 1.00$ on the 3D cantilever. Full acceptance means the safeguard was present and its criterion was met—not that the safeguard was inoperative. The clearest evidence of safeguard selectivity comes from the ablation studies. In 3D Ablation A (Table 5.5), setting $f_{\text{verify}} = 10$ collapses M3 acceptance to $\hat{\kappa} = 0.20$ while final compliance remains much closer to M1 than the M2 baseline ($\bar{C} = 12.705$ versus 12.331 at $f_{\text{verify}} = 5$). In 3D Ablation B (Table 5.6), tightening ϵ_{rel} to 0.005 drops M4 acceptance to $\hat{\kappa} = 0.55$ and raises compliance by 0.6 %, while M3 compliance is unchanged. Taken together, these results confirm that varying the verification parameters substantially alters acceptance behavior while final compliance remains far above the M2 degradation level. The main-table results establish that the default parameters prevent quality degradation; the ablation results establish that the mechanism is genuinely selective under appropriate conditions.

A modest verification frequency ($f_{\text{verify}} = 5$) and a lenient default tolerance ($\kappa_{\text{tr}} = 0.10$ in 2D, $\epsilon_{\text{rel}} = 0.01$ in 3D) are sufficient to prevent the compliance degradation

characteristic of coarse-only approaches. This is the central reliability finding within the tested benchmark family.

RQ3 — Robustness

Is the framework robust across the SIMP-based benchmark variants considered in this study?

The framework is robust within the tested family of static compliance-minimization benchmarks. M3 delivers consistent speedup and bounded compliance gap across all five benchmark problems, without problem-specific re-tuning of the acceptance parameters. The 2D ablations further confirm that M3 performance is largely insensitive to κ_{tr} over $[0.05, 0.20]$ and to f_{verify} over $\{3, 5, 10\}$ in 2D, with compliance changes of less than $0.05\,\%$ across the full parameter range.

The 3D results are more geometry-sensitive. On the cantilever, M3 and M4 both achieve speedup with compliance close to M1. On the L-bracket, M3 is slower than M1 at the default $f_{\text{verify}} = 5$ setting. The cantilever and the L-bracket do not reward the same method under the same parameters. This is a genuine limitation of the present study's scope, not a failure of the framework design. M4 achieved speedup on both tested 3D geometries without additional parameter tuning, although this did not translate into a uniformly better mean speed–quality tradeoff than M3.

Robustness here means stable behavior within the tested benchmark family. It does not mean universally applicable performance across all topology optimization formulations. The present results do not establish robustness for stress-constrained problems, multiple load cases, manufacturing constraints, or substantially different 3D geometries and mesh hierarchies. These extensions require separate evaluation.

RQ4 — Machine Learning Contribution

When ML-assisted initialization is integrated into the verified multifidelity workflow, does it improve initialization quality, acceptance behavior, or wall-clock time without compromising reliability, and under what verification-schedule conditions do these improvements translate into end-to-end runtime savings?

Regarding initialization quality, the U-Net provides a structured coarse-mesh starting field (mean predicted coarse compliance ≈49.1 on the cantilever, ≈55.9 on the L-bracket) rather than a uniform density. However, the number of coarse iterations to the first accepted verification event remains four across all seeds and both problems, indicating that the warm-start does not produce an earlier first acceptance at the present dataset scale.

The warm-start does measurably improve acceptance behavior on the cantilever: M4 achieves a mean acceptance rate of 1.00 versus 0.83 for M3 (seed-wise: 0.5/1.0/1.0 for

M3, 1.0/1.0/1.0 for M4). On the L-bracket both methods reach 1.00, so no differential is observable there.

Whether this translates into wall-clock savings depends on geometry. On the cantilever, M4 is slower than M3 (92.8 s versus 79.7 s) because Stage 1 runs longer despite the higher acceptance rate. On the L-bracket, M4 is 1.88× faster (62.6 s versus 117.7 s) because the warm-start shortens the coarse loop substantially. Reliability is preserved in both cases: no M4 run approaches M2-level degradation.

On the cantilever, σ_C falls from 0.276 (M3) to 0.027 (M4)—a 90 % reduction—suggesting that the warm-start narrows the explored design region. The acceptance rate $\hat{\kappa}$ serves as the practical diagnostic— when M4's $\hat{\kappa}$ matches or exceeds M3's, the warm-start contributes positively; otherwise M3 is the safer choice.

6.3 Limitations

The study is bounded by its experimental scope. Every reported speedup, compliance gap, and acceptance rate applies to the tested benchmark family: static, single-objective, single-load compliance minimization with the SIMP material model, a fixed filter radius, and a 4 : 1 element-count coarse-to-fine reduction ratio. The present framework assumes linear elastic material behaviour and small deformations throughout; consequently, the acceptance criterion—which compares fine-mesh compliance values under these assumptions—is not directly transferable to problems involving geometric or material nonlinearity.

In geometrically nonlinear settings [19, 20], the compliance surface is non-convex in a fundamentally different way, and the coarse-to-fine fidelity gap may grow substantially as the design evolves, making the calibrated acceptance thresholds used here unreliable without re-derivation. Similarly, multiphysics problems—such as coupled thermal–structural or fluid–topology formulations [21, 22]—introduce additional residuals that the present scalar compliance criterion does not capture; acceptance logic would need to be reformulated as a multi-objective or weighted-sum fidelity check. Extension to stress-constrained, multi-load, or manufacturing-constrained problems also requires modification of both the acceptance logic and the density update pipeline $\boldsymbol{\rho} \to \tilde{\boldsymbol{\rho}} \to \bar{\boldsymbol{\rho}}$. Whether the acceptance rates and compliance gaps observed here persist under any of these richer problem classes cannot be established from the present experiments and remains an open question for future investigation.

The statistical scope in 3D is limited. The main 3D experiments use $n = 3$ seeds per method–problem cell, and several ablation settings use $n = 2$ seeds or a single exploratory probe. With $n = 3$, the minimum achievable one-tailed Wilcoxon p-value is 0.125, which precludes formal hypothesis rejection at $\alpha = 0.05$. The 3D compliance comparisons, and particularly the M3 versus M4 comparison, are directional evidence in

the present sample rather than statistically resolved general conclusions. This applies especially to the L-bracket Ablation A2, which is based on a single seed and should be treated as indicative rather than definitive.

The ML warm-start is subject to a generalization limitation. Separate U-Net models were trained for the 3D cantilever and L-bracket problems, each on a relatively small problem-specific training corpus. The present experiments therefore evaluate the warm-start under a proof-of-concept regime. The observed variance-reduction benefit on the cantilever (σ_C reduced by approximately 90 %) reflects behavior within a single geometry and load family. Whether this benefit persists under broader geometry variation, different boundary conditions, or substantially out-of-distribution meshes is an open question. The current results support safe integration of learned proposals, not broad generalization across problem types.

Runtime performance depends on implementation details that are not universal. The 3D speedups are substantially lower than their 2D counterparts. This is mainly attributable to the PCG linear solver cost for 3D fine-mesh FEA: on the tested hardware, fine-mesh verification solves constitute a large fraction of total runtime, reducing the relative benefit of coarse-loop cycling. A more efficient 3D solver—multigrid-preconditioned or GPU-accelerated—would alter the coarse-to-fine cost ratio and could yield higher MFAS speedups without any algorithmic change. The reported values should therefore be understood as hardware- and solver-specific, not as fixed performance ceilings.

The current study explored selected fixed verification schedules and acceptance thresholds. The chosen values ($f_{\text{verify}} = 5$, $\kappa_{\text{tr}} = 0.10$, $\epsilon_{\text{rel}} = 0.01$) are empirically effective defaults within the tested range. They are not claimed to be universally optimal. Adaptive or theoretically grounded verification policies—which adjust f_{verify} based on coarse-loop convergence behavior—were not investigated. The ablation evidence suggests that the optimal f_{verify} is geometry-dependent: values that work well for the cantilever cause acceptance collapse on the L-bracket at the same setting. A problem-adaptive schedule is an identifiable area for further development.

6.4 Future Work

6.4.1 Algorithmic Extensions

The ablation evidence shows that f_{verify} is the most consequential tuning parameter in the framework. A natural extension is an adaptive verification schedule that adjusts f_{verify} dynamically based on observable convergence signals—for example, the rate of change of the coarse compliance or the gradient norm. Such a schedule could use sparse verification in the early, rapidly-changing phase and switch to frequent checking

near coarse convergence. The appropriate validation metric is the Pareto frontier of speedup versus compliance gap on the existing benchmark set, compared against the fixed schedules tested here.

Extending the acceptance criterion to multiple load cases would broaden the framework substantially. Both κ and ϵ_{rel} are currently scalar quantities defined for a single load vector. A generalization to a weighted-sum or minimax compliance over multiple loads requires a revised fidelity-gap analysis to confirm that acceptance rates comparable to those observed here remain achievable.

Richer rollback policies are also worth exploring. The present implementation restores the last accepted state on rejection. A trust-region-style policy could instead adjust the coarse proposal step size or temporarily increase f_{verify} in response to a rejection event, potentially recovering faster without the full cost of reverting to an earlier iterate.

A systematic study of the coarse-to-fine mesh-count ratio would quantify how the compliance gap and speedup trade off across different fidelity levels. The 4:1 ratio used throughout this work is a single point on that surface. Evaluating 2:1 and 8:1 ratios on the existing benchmark problems would require no new benchmark design.

6.4.2 Solver and Implementation Scaling

The 3D speedups observed in this study are bounded by the relative cost of fine-mesh PCG verification solves. Replacing the PCG solver with a multigrid-preconditioned or GPU-native solver would alter the coarse-to-fine cost ratio directly. The appropriate validation is to re-run the existing 3D benchmark matrix with the alternative solver while holding all MFAS algorithmic parameters fixed, and to compare the resulting speedup profile against Table 5.2.

The 3D fine meshes in this study contain 475,200 elements (cantilever: $180 \times 60 \times 44$) and 243,000 elements (L-bracket: $90 \times 90 \times 30$). Industrial topology optimization problems regularly involve 10^6 to 10^7 elements on distributed hardware. Whether the acceptance rates and compliance gaps reported here persist at substantially higher resolutions is an important open empirical question raised by this work. It is possible that the coarse-to-fine fidelity gap grows with problem scale in ways that reduce $\hat{\kappa}$ or make coarse-loop proposals systematically less informative. This cannot be resolved from the present experiments.

Stage-wise cost balancing is also worth investigating. The present results show that Stage 1 coarse-loop length—not Stage 0 inference overhead—is the primary determinant of M4 wall-clock time. Future work should therefore focus on reducing Stage 1 cost: tighter termination criteria, warm-started linear solvers, or coarse-mesh approximations with higher fidelity at low computational overhead.

6.4.3 Learning and Data Extensions

The current U-Net warm-start models were trained on relatively small, problem-specific datasets. The most direct extension is to train on larger and more geometrically diverse 3D corpora spanning multiple boundary condition families, load orientations, and structural archetypes. The primary evaluation criterion should be whether an increase in training diversity improves $\hat{\kappa}$ on held-out geometries while preserving the observed variance-reduction benefit.

An uncertainty-aware prediction head could assign a quality score to each U-Net proposal before it enters the MFAS loop. Low-confidence proposals could be deprioritized or replaced by a classical initialization, effectively creating a learned switching rule between M3 and M4 on a per-run basis. This would make M4 adaptive rather than fixed.

Online adaptation is another direction. A scheme that refines U-Net weights along the current optimization trajectory could progressively close the acceptance gap for problem instances distant from the training distribution. The appropriate success criterion is whether online-adapted M4 acceptance rates converge toward M3 baseline levels as a function of the number of online gradient steps.

Architecture alternatives beyond the current U-Net—including graph neural networks operating on the mesh topology, or equivariant networks that respect boundary-condition symmetries—could improve generalization without requiring exhaustive data augmentation.

6.4.4 Evaluation and Benchmarking Extensions

Extending the 3D seed ensemble from $n = 3$ to $n \geq 10$ per method–problem cell is the most immediate priority. It would provide the statistical power needed to formally distinguish compliance differences between M3 and M4, and to generate confidence intervals narrow enough to be meaningful for engineering tolerance analysis. This extension requires no algorithmic change.

Broader benchmark coverage would test external validity more rigorously. Future studies should include problems with multiple loads, passive domain constraints of varying complexity, frequency constraints, and geometries with re-entrant features more severe than the L-bracket. Comparisons with additional baselines—warm-started fine-mesh optimization without coarse cycling, and other learned initializers reviewed in Chapter 2—would place the MFAS speedup within the broader landscape of acceleration strategies.

Open benchmark release is also a priority. Releasing the mesh hierarchies, seed tables, configuration files, and result archives as a public dataset would enable external reproducibility checks and fill a gap that currently limits objective comparison of

acceleration methods in the topology optimization literature.

6.5 Conclusion

This dissertation addressed a practical question in density-based topology optimization: can coarse-mesh computational savings be captured reliably, with the quality of the fine-mesh result remaining quantifiable and controlled throughout? The experimental record in Chapters 4 and 5 answers this affirmatively within the tested scope. The MFAS framework achieves meaningful wall-clock reduction relative to fine-mesh SIMP, and the acceptance safeguard makes the associated compliance cost both bounded and transparent.

M3 proved to be the most reliable overall speed–quality compromise in this study. In 2D it delivers 2.13–2.30× speedup with compliance gaps below 0.80 % across all benchmarks and all seeds. In 3D it achieves 1.64× speedup on the cantilever with compliance close to M1 in the tested sample. The framework is straightforward to deploy: it requires no trained surrogate and no problem-specific parameter tuning beyond the recommended defaults.

M4 showed that learned warm-starting can be incorporated safely into the same verified pipeline. The verification mechanism applies identically to both M3 and M4, so the quality guarantee does not depend on prediction accuracy. At the dataset scales evaluated here, M4 does not consistently improve mean compliance or reduce wall-clock time relative to M3. It does, however, substantially reduce run-to-run compliance variability on the 3D cantilever—a benefit that is observable and potentially useful in reproducibility-sensitive workflows. Whether this benefit generalizes beyond the present benchmarks requires further investigation.

The defining element of the approach is the acceptance safeguard. It transforms coarse acceleration from an uncontrolled approximation into a bounded and auditable computation. The speedup becomes reproducible. The compliance cost becomes quantifiable. These properties, rather than any particular speedup number, constitute the principal contribution of the framework.

The conclusions apply to the tested benchmark family and implementation setting. They motivate continued development of verification-guided multifidelity methods at larger problem scales and under a broader range of structural objectives.

Bibliography

[1] J. D. Deaton and R. V. Grandhi, "A survey of structural and multidisciplinary continuum topology optimization: Post 2000," *Structural and Multidisciplinary Optimization*, vol. 49, no. 1, pp. 1–38, 2014. DOI: 10.1007/s00158-013-0956-z.

[2] G. I. N. Rozvany, "A critical review of established methods of structural topology optimization," *Structural and Multidisciplinary Optimization*, vol. 37, no. 3, pp. 217–237, 2009. DOI: 10.1007/s00158-007-0217-0.

[3] M. P. Bendsøe and N. Kikuchi, "Generating optimal topologies in structural design using a homogenization method," *Computer methods in applied mechanics and engineering*, vol. 71, no. 2, pp. 197–224, 1988. DOI: 10.1016/0045-7825(88)90086-2.

[4] M. P. Bendsøe, "Optimal shape design as a material distribution problem," *Structural Optimization*, vol. 1, no. 4, pp. 193–202, 1989. DOI: 10.1007/BF01650949.

[5] S. Mukherjee, D. Lu, B. Raghavan, P. Breitkopf, et al., "Accelerating large-scale topology optimization: State-of-the-art and challenges," *Archives of Computational Methods in Engineering*, 2021. DOI: 10.1007/s11831-021-09544-3.

[6] O. Sigmund, "A 99 line topology optimization code written in matlab," *Structural and Multidisciplinary Optimization*, vol. 21, no. 2, pp. 120–127, 2001. DOI: 10.1007/s001580050176.

[7] O. Sigmund and J. Petersson, "Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima," *Structural Optimization*, vol. 16, no. 1, pp. 68–75, 1998. DOI: 10.1007/BF01214002.

[8] J. K. Guest, "Topology optimization with multiple phase projection," *Computer Methods in Applied Mechanics and Engineering*, vol. 199, no. 1–4, pp. 123–135, 2009. DOI: 10.1016/j.cma.2009.09.023.

[9] L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders, *Large-Scale PDE-Constrained Optimization* (Lecture Notes in Computational Science and Engineering). Springer, 2003, vol. 30. DOI: 10.1007/978-3-642-55508-4.

[10] M. P. Bendsøe and O. Sigmund, *Topology Optimization: Theory, Methods, and Applications*. Berlin, Heidelberg: Springer, 2003. DOI: 10.1007/978-3-662-05086-6.

[11] K. Yaji, S. Yamasaki, and K. Fujita, "Multifidelity design guided by topology optimization," *Structural and Multidisciplinary Optimization*, vol. 61, no. 3, pp. 1071–1085, 2020. DOI: 10.1007/s00158-019-02406-4.

[12] Y. Yu, T. Hur, J. Jung, and I. G. Jang, "Deep learning for determining a near-optimal topological design without any iteration," *Structural and Multidisciplinary Optimization*, vol. 59, no. 3, pp. 787–799, 2019. DOI: 10.1007/s00158-018-2101-5.

[13] A. Chandrasekhar and K. Suresh, "TOuNN: Topology optimization using neural networks," *Structural and Multidisciplinary Optimization*, vol. 63, no. 3, pp. 1135–1149, 2021. DOI: 10.1007/s00158-020-02748-4.

[14] S. Shin, D. Shin, and N. Kang, "Topology optimization via machine learning and deep learning: A review," *Journal of Computational Design and Engineering*, vol. 10, no. 4, pp. 1736–1766, 2023. DOI: 10.1093/jcde/qwad072.

[15] R. V. Woldseth, N. Aage, J. A. Bærentzen, and O. Sigmund, "On the use of artificial neural networks in topology optimisation," *Structural and Multidisciplinary Optimization*, vol. 65, no. 10, p. 294, 2022. DOI: 10.1007/s00158-022-03347-1.

[16] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, and O. Sigmund, "Efficient topology optimization in MATLAB using 88 lines of code," *Structural and Multidisciplinary Optimization*, vol. 43, no. 1, pp. 1–16, 2011. DOI: 10.1007/s00158-010-0594-7.

[17] N. Tatke and J. Kaczmarczyk, "Multifidelity topology optimization with runtime verification and acceptance control: Benchmark study in 2d and 3d," *Materials*, vol. 19, no. 4, p. 769, 2026. DOI: 10.3390/ma19040769.

[18] O. Sigmund, "On the design of compliant mechanisms using topology optimization*," *Mechanics of Structures and Machines*, vol. 25, no. 4, pp. 493–524, 1997. DOI: 10.1080/08905459708945415.

[19] T. Buhl, C. B. W. Pedersen, and O. Sigmund, "Stiffness design of geometrically nonlinear structures using topology optimization," *Structural and Multidisciplinary Optimization*, vol. 19, no. 2, pp. 93–104, 2000. DOI: 10.1007/s001580050089.

[20] T. E. Bruns and D. A. Tortorelli, "Topology optimization of non-linear elastic structures and compliant mechanisms," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 26–27, pp. 3443–3459, 2001. DOI: 10.1016/S0045-7825(00)00278-4.

[21] T. Borrvall and J. Petersson, "Topology optimization of fluids in Stokes flow," *International Journal for Numerical Methods in Fluids*, vol. 41, no. 1, pp. 77–107, 2003. DOI: 10.1002/fld.426.

[22] X. Qian and E. M. Dede, "Topology optimization of a coupled thermal-fluid system under a tangential thermal gradient constraint," *Structural and Multidisciplinary Optimization*, vol. 54, no. 3, pp. 531–551, 2016. DOI: 10.1007/s00158-016-1421-6.

[23] M. Zhou and G. I. Rozvany, "The COC algorithm, part II: Topological, geometrical and generalized shape optimization," *Computer methods in applied mechanics and engineering*, vol. 89, no. 1–3, pp. 309–336, 1991. DOI: 10.1016/0045-7825(91)90046-9.

[24] M. Y. Wang, X. Wang, and D. Guo, "A level set method for structural topology optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 192, no. 1–2, pp. 227–246, 2003. DOI: 10.1016/S0045-7825(02)00559-5.

[25] N. P. Van Dijk, K. Maute, M. Langelaar, and F. Van Keulen, "Level-set methods for structural topology optimization: A review," *Structural and Multidisciplinary Optimization*, vol. 48, no. 3, pp. 437–472, 2013. DOI: 10.1007/s00158-013-0912-y.

[26] C. S. Andreasen, M. O. Elingaard, and N. Aage, "Level set topology and shape optimization by density methods using cut elements with length scale control," *Structural and Multidisciplinary Optimization*, vol. 62, no. 2, pp. 685–707, 2020. DOI: 10.1007/s00158-020-02527-1.

[27] T. Zhang, X. Yang, and X. Wang, "Level set-based topology optimization for thermal-fluid system based on the radial basis functions," *Applied Mathematical Modelling*, vol. 113, pp. 144–159, 2023. DOI: 10.1016/j.apm.2022.09.005.

[28] B. Bourdin and A. Chambolle, "Design-dependent loads in topology optimization," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 9, pp. 19–48, 2003. DOI: 10.1051/cocv:2002070.

[29] A. Takezawa, S. Nishiwaki, and M. Kitamura, "Shape and topology optimization based on the phase field method and sensitivity analysis," *Journal of Computational Physics*, vol. 229, no. 7, pp. 2697–2718, 2010. DOI: 10.1016/j.jcp.2009.12.017.

[30] Y. M. Xie and G. P. Steven, "A simple evolutionary procedure for structural optimization," *Computers & structures*, vol. 49, no. 5, pp. 885–896, 1993. DOI: 10.1016/0045-7949(93)90035-C.

[31] P. Hajela and E. Lee, "Genetic algorithms in truss topological optimization," *International Journal of Solids and Structures*, vol. 32, no. 22, pp. 3341–3357, 1995. DOI: 10.1016/0020-7683(94)00306-H.

[32] R. E. Perez and K. Behdinan, "Particle swarm approach for structural design optimization," *Computers & Structures*, vol. 85, no. 19–20, pp. 1579–1588, 2007. DOI: 10.1016/j.compstruc.2006.10.013.

[33] H. Rahami, A. Kaveh, and Y. Gholipour, "Sizing, geometry and topology optimization of trusses via force method and genetic algorithm," *Engineering Structures*, vol. 30, no. 9, pp. 2360–2369, 2008. DOI: 10.1016/j.engstruct.2008.01.012.

[34] M. Sönmez, "Artificial bee colony algorithm for optimization of truss structures," *Applied Soft Computing*, vol. 11, no. 2, pp. 2406–2418, 2011. DOI: 10.1016/j.asoc.2010.09.003.

[35] V. Ho-Huu, T. Nguyen-Thoi, M. H. Nguyen-Thoi, and L. Le-Anh, "An improved constrained differential evolution using discrete variables (d-icde) for layout optimization of truss structures," *Expert Systems with Applications*, vol. 42, no. 20, pp. 7057–7069, 2015. DOI: 10.1016/j.eswa.2015.04.072.

[36] M. Abd Elaziz, A. H. Elsheikh, D. Oliva, L. Abualigah, S. Lu, and A. A. Ewees, "Advanced metaheuristic techniques for mechanical design problems: Review," *Archives of Computational Methods in Engineering*, vol. 29, no. 1, pp. 695–716, 2022. DOI: 10.1007/s11831-021-09589-4.

[37] H. Lin, H. Liu, and P. Wei, "A parallel parameterized level set topology optimization framework for large-scale structures with unstructured meshes," *Computer Methods in Applied Mechanics and Engineering*, vol. 397, p. 115 112, 2022. DOI: 10.1016/j.cma.2022.115112.

[38] G. Allaire, F. Jouve, and A.-M. Toader, "Structural optimization using sensitivity analysis and a level-set method," *Journal of Computational Physics*, vol. 194, no. 1, pp. 363–393, 2004. DOI: 10.1016/j.jcp.2003.09.032.

[39] M. Stolpe and K. Svanberg, "An alternative interpolation scheme for minimum compliance topology optimization," *Structural and Multidisciplinary Optimization*, vol. 22, no. 2, pp. 116–124, 2001. DOI: 10.1007/s001580100129.

[40] S. Rojas-Labanda, O. Sigmund, and M. Stolpe, "A short numerical study on the optimization methods influence on topology optimization," *Structural and Multidisciplinary Optimization*, vol. 56, no. 6, pp. 1603–1612, 2017. DOI: 10.1007/s00158-017-1813-2.

[41] K. Svanberg, "The method of moving asymptotes—a new method for structural optimization," *International Journal for Numerical Methods in Engineering*, vol. 24, no. 2, pp. 359–373, 1987. DOI: 10.1002/nme.1620240207.

[42] B. S. Lazarov, M. Schevenels, and O. Sigmund, "Topology optimization with geometric uncertainties by perturbation techniques," *International Journal for Numerical Methods in Engineering*, vol. 90, no. 11, pp. 1321–1336, 2012. DOI: https://doi.org/10.1002/nme.3361.

[43] B. Bourdin, "Filters in topology optimization," *International Journal for Numerical Methods in Engineering*, vol. 50, no. 9, pp. 2143–2158, 2001. DOI: 10.1002/nme.116.

[44] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-based analysis and optimization," *Progress in aerospace sciences*, vol. 41, no. 1, pp. 1–28, 2005. DOI: 10.1016/j.paerosci.2005.02.001.

[45] I. Negrin, M. Kripka, and V. Yepes, "Metamodel-assisted design optimization in the field of structural engineering: A literature review," *Structures*, vol. 52, pp. 609–631, 2023. DOI: 10.1016/j.istruc.2023.04.006.

[46] E. Iuliano, "Global optimization of benchmark aerodynamic cases using physics-based surrogate models," *Aerospace Science and Technology*, vol. 67, pp. 273–286, 2017. DOI: 10.1016/j.ast.2017.04.013.

[47] A. J. Keane and I. I. Voutchkov, "Robust design optimization using surrogate models," *Journal of Computational Design and Engineering*, vol. 7, no. 1, pp. 44–55, 2020. DOI: 10.1093/jcde/qwaa005.

[48] A. I. Forrester and A. J. Keane, "Recent advances in surrogate-based optimization," *Progress in aerospace sciences*, vol. 45, no. 1, pp. 50–79, 2009. DOI: 10.1016/j.paerosci.2008.11.001.

[49] M. A. Bouhlel and J. R. R. A. Martins, "Gradient-enhanced kriging for high-dimensional problems," *Engineering with Computers*, vol. 35, no. 1, pp. 157–173, 2019. DOI: 10.1007/s00366-018-0590-x.

[50] F. Ferrari and O. Sigmund, "A new generation 99 line Matlab code for compliance topology optimization and its extension to 3D," *Structural and Multidisciplinary Optimization*, vol. 62, no. 4, pp. 2211–2228, 2020. DOI: 10.1007/s00158-020-02629-w.

[51] K. Liu and A. Tovar, "An efficient 3d topology optimization code written in MATLAB," *Structural and Multidisciplinary Optimization*, vol. 50, no. 6, pp. 1175–1196, 2014. DOI: 10.1007/s00158-014-1107-x.

[52] Z. H. Zuo and Y. M. Xie, "A simple and compact python code for complex 3D topology optimization," *Advances in Engineering Software*, vol. 85, pp. 1–11, 2015. DOI: 10.1016/j.advengsoft.2015.02.006.

[53] A. Gupta, R. Chowdhury, A. Chakrabarti, and T. Rabczuk, "A 55-line code for large-scale parallel topology optimization in 2d and 3d," *arXiv*, 2020. DOI: 10.48550/arXiv.2012.08208.

[54] R. M. Ferro and R. Pavanello, "A simple and efficient structural topology optimization implementation using open-source software for all steps of the algorithm: Modeling, sensitivity analysis and optimization," *Computer Modeling in Engineering & Sciences*, vol. 136, no. 2, pp. 1371–1397, 2023. DOI: 10.32604/cmes.2023.026043.

[55] J. Yan, R. Xiang, D. Kamensky, M. T. Tolley, and J. T. Hwang, "Topology optimization with automated derivative computation for multidisciplinary design problems," *Structural and Multidisciplinary Optimization*, vol. 65, no. 5, 2022. DOI: 10.1007/s00158-022-03168-2.

[56] Y. Wang, X. Li, K. Long, and P. Wei, "Open-source codes of topology optimization: A summary for beginners to start their research," *Computer Modeling in Engineering & Sciences*, vol. 137, no. 1, pp. 1–34, 2023. DOI: 10.32604/cmes.2023.027603.

[57] R. Merli, A. Martínez-Martínez, J. Ródenas, M. Bosch-Galera, and E. Nadal, "Two-level high-resolution structural topology optimization with equilibrated cells," *Computer-Aided Design*, 2025. DOI: 10.1016/j.cad.2024.103811.

[58] E. de Sturler, G. H. Paulino, and S. Wang, "Topology optimization with adaptive mesh refinement," Virginia Tech, Tech. Rep., 2008.

[59] J. Martínez-Frutos, P. J. Martínez-Castejón, and D. Herrero-Pérez, "Efficient topology optimization using gpu computing with multilevel granularity," *Advances in Engineering Software*, vol. 106, pp. 47–62, 2017. DOI: 10.1016/j.advengsoft.2017.01.009.

[60] M. Appel et al., "Space-time multigrid methods suitable for topology optimization," *arXiv*, 2025. DOI: 10.48550/arXiv.2505.10168.

[61] S. Wang, E. de Sturler, and G. H. Paulino, "Dynamic adaptive mesh refinement for topology optimization," *arXiv*, 2010. DOI: 10.48550/arXiv.1009.4975.

[62] T. H. Nguyen, G. H. Paulino, J. Song, and C. H. Le, "Topology optimization using the p-version of the finite element method," *Structural and Multidisciplinary Optimization*, vol. 41, no. 4, pp. 605–618, 2010. DOI: https://doi.org/10.1007/s00158-017-1675-7.

[63] J. C. A. Costa Jr. and M. K. Alves, "Layout optimization with h-adaptivity of structures," *International Journal for Numerical Methods in Engineering*, vol. 58, no. 1, pp. 83–102, 2003. DOI: 10.1002/nme.759.

[64] A. I. J. Forrester, A. Sóbester, and A. J. Keane, "Multi-fidelity optimization via surrogate modelling," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, pp. 3251–3269, 2007. DOI: 10.1098/rspa.2007.1900.

[65] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Survey of multifidelity methods in uncertainty propagation, inference, and optimization," *SIAM Review*, vol. 60, no. 3, pp. 550–591, 2018. DOI: 10.1137/16M1082469.

[66] S. G. Kontogiannis and M. A. Savill, "A generalized methodology for multi-disciplinary design optimization using surrogate modelling and multifidelity analysis," *Optimization and Engineering*, vol. 21, no. 3, pp. 723–759, 2020. DOI: 10.1007/s11081-020-09504-z.

[67] H. Kawabe, K. Yaji, and Y. Aoki, "Data-driven multifidelity topology design with multi-channel variational auto-encoder for concurrent optimization of multiple design variable fields," *Computer Methods in Applied Mechanics and Engineering*, vol. 437, p. 117 772, 2025. DOI: 10.1016/j.cma.2025.117772.

[68] R. J. Gladstone, M. A. Nabian, V. Keshavarzzadeh, and H. Meidani, "Robust topology optimization using multi-fidelity variational autoencoders," *Journal of Machine Learning for Modeling and Computing*, vol. 5, no. 4, pp. 23–52, 2024. DOI: 10.1615/JMachLearnModelComput.2024054646.

[69] S. Yamasaki, K. Yaji, and K. Fujita, "Data-driven topology design using a deep generative model," *Structural and Multidisciplinary Optimization*, vol. 64, no. 3, pp. 1401–1420, 2021. DOI: 10.1007/s00158-021-02926-y.

[70] B. Klein and M. Ohlberger, "Multi-fidelity learning of reduced order models for parabolic pde constrained optimization," *arXiv*, 2025. DOI: 10.48550/arXiv.2503.21252.

[71] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2000. DOI: 10.1137/1.9780898719857.

[72] J. F. Rodríguez, J. E. Renaud, L. T. Watson, B. A. Wujek, and R. V. Tappeta, "Trust region model management in multidisciplinary design optimization," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1–2, pp. 139–154, 2000. DOI: 10.1016/S0377-0427(00)00424-6.

[73] N. Alexandrov, J. E. Dennis, R. M. Lewis, and V. Torczon, "A trust region framework for managing the use of approximation models in optimization," *Structural Optimization*, vol. 15, no. 1, pp. 16–23, 1998. DOI: 10.1007/BF01197433.

[74] N. M. Alexandrov, R. M. Lewis, C. R. Gumbert, L. L. Green, and P. A. Newman, "Approximation and model management in aerodynamic optimization with variable-fidelity models," *Journal of Aircraft*, vol. 38, no. 6, pp. 1093–1101, 2001. DOI: 10.2514/2.2877.

[75] A. I. J. Forrester, N. W. Bressloff, and A. J. Keane, "Optimization using surrogate models and partially converged computational fluid dynamics simulations," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 462, no. 2071, pp. 2177–2204, 2006. DOI: 10.1098/rspa.2006.1679.

[76] E. Qian, M. Grepl, K. Veroy, and K. Willcox, "A certified trust region reduced basis approach to PDE-constrained optimization," *SIAM Journal on Scientific Computing*, vol. 39, no. 5, S434–S460, 2017. DOI: 10.1137/16M1081981.

[77] T. Keil, L. Mechelli, M. Ohlberger, F. Schindler, and S. Volkwein, "A non-conforming dual approach for adaptive trust-region reduced basis approximation of PDE-constrained parameter optimization," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 55, no. 3, pp. 1239–1269, 2021. DOI: 10.1051/m2an/2021019.

[78] M. Kartmann, T. Keil, M. Ohlberger, S. Volkwein, and B. Kaltenbacher, "Adaptive reduced basis trust region methods for parameter identification problems," *Computational Science and Engineering*, vol. 1, no. 1, p. 3, 2024. DOI: 10.1007/s44207-024-00002-z.

[79] K. Yaji, S. Yamasaki, and K. Fujita, "Data-driven multifidelity topology design using a deep generative model: Application to forced convection heat transfer problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 388, p. 114 284, 2022. DOI: 10.1016/j.cma.2021.114284.

[80] J. K. Guest, J. H. Prévost, and T. Belytschko, "Achieving minimum length scale in topology optimization using nodal design variables and projection functions," *International Journal for Numerical Methods in Engineering*, vol. 61, no. 2, pp. 238–254, 2004. DOI: 10.1002/nme.1064.

[81] F. Wang, B. S. Lazarov, and O. Sigmund, "On projection methods, convergence and robust formulations in topology optimization," *Structural and Multidisciplinary Optimization*, vol. 43, pp. 767–784, 2011. DOI: 10.1007/s00158-010-0602-y.

[82] R. Becker and R. Rannacher, "An optimal control approach to a posteriori error estimation in finite element methods," *Acta Numerica*, vol. 10, pp. 1–102, 2001. DOI: 10.1017/S0962492901000010.

[83] M. Bruggi and M. Verani, "A fully adaptive topology optimization algorithm with goal-oriented error control," *Computers & Structures*, vol. 89, no. 15–16, pp. 1481–1493, 2011. DOI: 10.1016/j.compstruc.2011.05.003.

[84] M. A. Salazar de Troya and D. A. Tortorelli, "Adaptive mesh refinement in stress-constrained topology optimization," *Structural and Multidisciplinary Optimization*, 2018. DOI: 10.1007/s00158-018-2084-2.

[85] A. March and K. Willcox, "Provably convergent multifidelity optimization algorithm not requiring high-fidelity derivatives," *AIAA Journal*, vol. 50, no. 5, pp. 1079–1089, 2012. DOI: 10.2514/1.J051125.

[86] P. Ramu, P. Thananjayan, E. Acar, G. Bayrak, J. W. Park, and I. Lee, "A survey of machine learning techniques in structural and multidisciplinary optimization," *Structural and Multidisciplinary Optimization*, vol. 65, no. 9, p. 266, 2022. DOI: 10.1007/s00158-022-03369-9.

[87] A. Chandrasekhar and K. Suresh, "Approximate length scale filter in topology optimization using fourier enhanced neural networks," *Computer-Aided Design*, vol. 150, p. 103 277, 2022. DOI: 10.1016/j.cad.2022.103277.

[88] H. Chen, A. Joglekar, and L. B. Kara, "Topology optimization using neural networks with conditioning field initialization for improved efficiency," *Journal of Mechanical Design*, vol. 146, no. 6, p. 061 702, 2023. DOI: 10.1115/1.4064131.

[89] Z. Zhang, Y. Li, W. Zhou, X. Chen, W. Yao, and Y. Zhao, "TONR: An exploration for a novel way combining neural network with topology optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 386, p. 114 083, 2021. DOI: 10.1016/j.cma.2021.114083.

[90] I. Kuszczak, G. Kuś, F. Bosi, and M. A. Bessa, "Meta-neural topology optimization: Knowledge infusion with meta-learning," *arXiv*, 2025. DOI: 10.48550/arXiv.2502.01830.

[91] H. T. Kollmann, D. W. Abueidda, S. Koric, E. Guleryuz, and N. A. Sobh, "Deep learning for topology optimization of 2D metamaterials," *Materials & Design*, vol. 196, p. 109 098, 2020. DOI: 10.1016/j.matdes.2020.109098.

[92] D. W. Abueidda, S. Koric, and N. A. Sobh, "Topology optimization of 2D structures with nonlinearities using deep learning," *Computers & Structures*, vol. 237, p. 106 283, 2020. DOI: 10.1016/j.compstruc.2020.106283.

[93] M. M. Islam and L. Liu, "Deep learning accelerated topology optimization with inherent control of image quality," *Structural and Multidisciplinary Optimization*, vol. 65, no. 11, p. 325, 2022. DOI: 10.1007/s00158-022-03433-4.

[94] J. Rasulzade, S. Rustamov, B. Akhmetov, Y. Maksum, and M. Nogaibayeva, "Computational acceleration of topology optimization using deep learning," *Applied Sciences*, vol. 13, no. 1, p. 479, 2023. DOI: 10.3390/app13010479.

[95] J. Zehnder, Y. Li, S. Coros, and B. Thomaszewski, "Ntopo: Mesh-free topology optimization using implicit neural representations," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 10 368–10 381.

[96] L. Herrmann, O. Sigmund, V. M. Li, C. Vogl, and S. Kollmannsberger, "On neural networks for generating better local optima in topology optimization," *Structural and Multidisciplinary Optimization*, vol. 67, no. 11, p. 192, 2024. DOI: 10.1007/s00158-024-03908-6.

[97] R. K. Tan, C. Qian, K. Li, D. Xu, and W. Ye, "An adaptive and scalable artificial neural network-based model-order-reduction method for large-scale topology optimization designs," *Structural and Multidisciplinary Optimization*, vol. 65, no. 12, p. 348, 2022. DOI: 10.1007/s00158-022-03456-x.

[98] A. M. Propp and D. M. Tartakovsky, "Transfer learning on multi-dimensional data: A novel approach to neural network-based surrogate modeling," *Journal of Machine Learning for Modeling and Computing*, vol. 6, no. 2, pp. 13–27, 2025. DOI: 10.1615/JMachLearnModelComput.2024057138.

[99] Z. Zhang, W. Yao, Y. Li, W. Zhou, and X. Chen, "Topology optimization via implicit neural representations," *Computer Methods in Applied Mechanics and Engineering*, vol. 411, p. 116 052, 2023. DOI: 10.1016/j.cma.2023.116052.

[100] A. P. Padhi, S. Chakraborty, A. Chakrabarti, and R. Chowdhury, "Deep learning accelerated efficient framework for topology optimization," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108 559, 2024. DOI: 10.1016/j.engappai.2024.108559.

[101] Z. Nie, T. Lin, H. Jiang, and L. B. Kara, "TopologyGAN: Topology optimization using generative adversarial networks based on physical fields over the initial domain," *Journal of Mechanical Design*, vol. 143, no. 3, p. 031 715, 2021. DOI: 10.1115/1.4049533.

[102] H. Kazemi, C. C. Seepersad, and H. Alicia Kim, "Multiphysics design optimization via generative adversarial networks," *Journal of Mechanical Design*, vol. 144, no. 12, p. 121 702, 2022. DOI: 10.1115/1.4055377.

[103] L. Pereira and L. Driemeier, "Wasserstein generative adversarial networks for topology optimization," *Structures*, vol. 67, p. 106 924, 2024. DOI: 10.1016/j.istruc.2024.106924.

[104] Q. Zeng, X. Liu, X. Zhu, X. Zhang, and P. Hu, "Data-driven structural topology optimization method using conditional wasserstein generative adversarial networks with gradient penalty," *CMES–Computer Modeling in Engineering & Sciences*, vol. 141, no. 3, pp. 2065–2085, 2024. DOI: 10.32604/cmes.2024.052620.

[105] A. Padmaprabhan et al., "GO-GAN: Geometry optimization generative adversarial network for achieving optimized structures with targeted physical properties," *arXiv*, 2025. DOI: 10.48550/arXiv.2502.00416.

[106] W. Zhang, G. Zhao, and L. Su, "Research on multi-stage topology optimization method based on latent diffusion model," *Advanced Engineering Informatics*, vol. 63, p. 102 966, 2025. DOI: 10.1016/j.aei.2024.102966.

[107] G. Giannone and F. Ahmed, "Diffusing the optimal topology: A generative optimization approach," *arXiv*, 2023. DOI: 10.48550/arXiv.2303.09760.

[108] Y. Gao, S. Zhou, and M. Li, "Structural topology optimization based on diffusion generative adversarial networks," *Engineering Applications of Artificial Intelligence*, vol. 138, p. 109 444, 2024. DOI: 10.1016/j.engappai.2024.109444.

[109] Y. Gao, S. Zhou, and M. Li, "Structural topology optimization based on deep learning," *Journal of Computational Physics*, vol. 520, p. 113 506, 2025. DOI: 10.1016/j.jcp.2024.113506.

[110] S. Oh, Y. Jung, S. Kim, I. Lee, and N. Kang, "Deep generative design: Integration of topology optimization and generative models," *Journal of Mechanical Design*, vol. 141, no. 11, p. 111 405, 2019. DOI: 10.1115/1.4044229.

[111] L. Regenwetter, A. H. Nobari, and F. Ahmed, "Deep generative models in engineering design: A review," *Journal of Mechanical Design*, vol. 144, no. 7, p. 071 704, 2022. DOI: 10.1115/1.4053859.

[112] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019. DOI: 10.1016/j.jcp.2018.10.045.

[113] J. Yin, Z. Wen, S. Li, Y. Zhang, and H. Wang, "Dynamically configured physics-informed neural network in topology optimization applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 426, p. 117 004, 2024. DOI: 10.1016/j.cma.2024.117004.

[114] H. Jeong, J. Bai, C. P. Batuwatta-Gamage, C. Rathnayaka, Y. Zhou, and Y. Gu, "A physics-informed neural network-based topology optimization (PINNTO) framework for structural optimization," *Engineering Structures*, vol. 278, p. 115 484, 2023. DOI: 10.1016/j.engstruct.2022.115484.

[115] H. Jeong et al., "A complete physics-informed neural network-based framework for structural topology optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 417, p. 116 401, 2023. DOI: 10.1016/j.cma.2023.116401.

[116] H. Jeong, C. Batuwatta-Gamage, J. Bai, C. Rathnayaka, Y. Zhou, and Y. Gu, "An advanced physics-informed neural network-based framework for nonlinear and complex topology optimization," *Engineering Structures*, vol. 322, p. 119 194, 2025. DOI: 10.1016/j.engstruct.2024.119194.

[117] A. Singh, S. Chakraborty, and R. Chowdhury, "A dual physics-informed neural network for topology optimization," *arXiv*, 2024. DOI: 10.48550/arXiv.2410.14342.

[118] J. He, C. Chadha, S. Kushwaha, S. Koric, D. Abueidda, and I. Jasiuk, "Deep energy method in topology optimization applications," *Acta Mechanica*, vol. 234, no. 4, pp. 1365–1379, 2023. DOI: 10.1007/s00707-022-03449-3.

[119] J. Yin, S. Li, Y. Zhang, and H. Wang, "An efficient discrete physics-informed neural networks for geometrically nonlinear topology optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 442, p. 118 043, 2025. DOI: 10.1016/j.cma.2025.118043.

[120] D. Erzmann and S. Dittmer, "Equivariant neural operators for gradient-consistent topology optimization," *Journal of Computational Design and Engineering*, vol. 11, no. 3, pp. 91–100, 2024. DOI: 10.1093/jcde/qwae039.

[121] K. Liang, D. Zhu, and F. Li, "A fourier neural operator-based lightweight machine learning framework for topology optimization," *Applied Mathematical Modelling*, vol. 129, pp. 714–732, 2024. DOI: 10.1016/j.apm.2024.02.011.

[122] K. Shukla et al., "Deep neural operators as accurate surrogates for shape optimization," *Engineering Applications of Artificial Intelligence*, vol. 129, p. 107 615, 2024. DOI: 10.1016/j.engappai.2023.107615.

[123] R. Olabiyi, H. Yang, and A. Iquebal, "CRONet: A convolutional recurrent operator approximator network to accelerate topology optimization," *Manufacturing Letters*, 2025. DOI: 10.1016/j.mfglet.2025.06.125.

[124] C. Wang, Z. Zhao, M. Zhou, O. Sigmund, and X. S. Zhang, "A comprehensive review of educational articles on structural and multidisciplinary optimization," *Structural and Multidisciplinary Optimization*, vol. 64, pp. 2827–2880, 2021. DOI: 10.1007/s00158-021-03050-7.

[125] J. N. Reddy, *An Introduction to the Finite Element Method*, 3rd ed. McGraw-Hill, 2006.

[126] S. C. Brenner and R. Scott, *The Mathematical Theory of Finite Element Methods* (Texts in Applied Mathematics), 3rd ed. Springer, 2008, vol. 15. DOI: 10.1007/978-0-387-75934-0.

[127] O. Sigmund, "Morphology-based black and white filters for topology optimization," *Structural and Multidisciplinary Optimization*, vol. 33, no. 4-5, pp. 401–424, 2007. DOI: 10.1007/s00158-006-0087-x.

[128] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*. Academic Press, 2001.

[129] Y. X. Yuan, "A review of trust region algorithms for optimization," in *ICIAM 99: Proceedings of the Fourth International Congress on Industrial & Applied Mathematics, Edinburgh*, J. M. Ball and J. C. R. Hunt, Eds., Oxford University Press, 2000, pp. 271–282. DOI: 10.1093/oso/9780198505143.003.0023.

[130] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *arXiv*, 2015. DOI: 10.48550/arXiv.1505.04597.

[131] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[132] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv*, 2015. DOI: 10.48550/arXiv.1502.03167.

[133] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

[134] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, 2015. DOI: 10.48550/arXiv.1412.6980.

Appendix A

Implementation Snippets: Two-Dimensional Framework

This appendix covers selected implementation excerpts for the two-dimensional MFAS framework, covering the FEM core (Sections A.1–A.2) and the MFAS-specific components (Sections A.3–A.4).

A.1 Q4 Element Stiffness and DOF Connectivity

```
def element_stiffness_q4(E=1.0, nu=0.3):
    k  = np.array([1/2-nu/6, 1/8+nu/8, -1/4-nu/12,
       -1/8+3*nu/8,
                   -1/4+nu/12,-1/8-nu/8,  nu/6,
                      1/8-3*nu/8])
    KE = E/(1-nu**2) * np.array([
         [k[0],k[1],k[2],k[3],k[4],k[5],k[6],k[7]],
         [k[1],k[0],k[7],k[6],k[5],k[4],k[3],k[2]],
         [k[2],k[7],k[0],k[5],k[6],k[3],k[4],k[1]],
         [k[3],k[6],k[5],k[0],k[7],k[2],k[1],k[4]],
         [k[4],k[5],k[6],k[7],k[0],k[1],k[2],k[3]],
         [k[5],k[4],k[3],k[2],k[1],k[0],k[7],k[6]],
         [k[6],k[3],k[4],k[1],k[2],k[7],k[0],k[5]],
         [k[7],k[2],k[1],k[4],k[3],k[6],k[5],k[0]]])
    return KE

def build_edof(nelx, nely):
    nn = (nelx+1)*(nely+1)
    nodenrs = np.arange(nn).reshape((nely+1, nelx+1),
       order='F')
```

```
    edofMat = np.zeros((nelx*nely, 8), dtype=int)
    for elx in range(nelx):
        for ely in range(nely):
            el = elx*nely + ely
            n1=nodenrs[ely,  elx];   n2=nodenrs[ely,  elx+1]
            n3=nodenrs[ely+1,elx+1]; n4=nodenrs[ely+1,elx]
            edofMat[el,:] = [2*n1,2*n1+1, 2*n2,2*n2+1,
                             2*n3,2*n3+1, 2*n4,2*n4+1]
    iK = np.kron(edofMat, np.ones((8,1),dtype=int)).flatten()
    jK = np.kron(edofMat, np.ones((1,8),dtype=int)).flatten()
    return edofMat, iK, jK, 2*nn, nodenrs

def build_filter(nelx, nely, rmin):
    H_i, H_j, H_v = [], [], []
    r = int(np.floor(rmin))
    for i in range(nelx):
        for j in range(nely):
            row = i*nely + j
            for ii in range(max(i-r,0), min(i+r,nelx-1)+1):
                for jj in range(max(j-r,0),
                    min(j+r,nely-1)+1):
                    w = rmin - np.sqrt((i-ii)**2+(j-jj)**2)
                    if w > 0:
                        H_i.append(row);
                            H_j.append(ii*nely+jj)
                        H_v.append(w)
    H  = sp.coo_matrix((H_v,(H_i,H_j)),
                       shape=(nelx*nely,nelx*nely)).tocsr()
    return H, np.asarray(H.sum(axis=1)).flatten()
```

Listing A.1: Q4 unit-modulus element stiffness (*top*) and element DOF connectivity with filter assembly (*bottom*). (Eqs. 3.6 and 3.13).

A.2 Compliance Evaluation, Sensitivity, and OC Update

```
def fea_compliance(nelx, nely, xPhys, problem,
                   KE, edofMat, iK, jK, nodenrs):
    E  = (problem.Emin +
       xPhys.flatten(order='F')**problem.penal
```

```
4           * (problem.E0 - problem.Emin))     # SIMP
                interpolation
5     sK = (KE.flatten()[:,None]*E[None,:]).flatten(order='F')
6     K  = sp.coo_matrix((sK,(iK,jK)),
7               shape=(2*(nelx+1)*(nely+1),)*2).tocsr()
8     K  = (K + K.T)*0.5
9     F, fixeddofs, _ = build_bcs(nelx, nely, nodenrs, problem)
10    freedofs = np.setdiff1d(np.arange(K.shape[0]), fixeddofs)
11    U = np.zeros(K.shape[0])
12    U[freedofs] = spla.spsolve(
13        K[np.ix_(freedofs,freedofs)], F[freedofs])
14    return float(F @ U), U, F, fixeddofs
15
16# --- Sensitivity and OC update (executed each iteration) ---
17ce = np.sum((U[edofMat] @ KE) * U[edofMat], axis=1)
18dc = (-problem.penal*(problem.E0-problem.Emin)
19      * xPhys.flatten(order='F')**(problem.penal-1)*ce
20     ).reshape((nely,nelx), order='F')
21x_flat = x.flatten(order='F')
22dc_f   = ((H @ (x_flat*dc.flatten(order='F')))
23          / (Hs*np.maximum(1e-3, x_flat))
24         ).reshape((nely,nelx), order='F')   # filtered
               sensitivity
25x      = oc_update(x, dc_f, volfrac, move=0.2)
26xPhys  = apply_filter(H, Hs, x)
27
28def oc_update(x, dc, volfrac, move=0.2, xmin=0.001, xmax=1.0):
29    l1, l2 = 0.0, 1e9
30    while (l2-l1)/(l2+l1+1e-12) > 1e-3:
31        lmid = 0.5*(l2+l1)
32        xnew = np.clip(
33            np.clip(x*np.sqrt(-dc/(lmid+1e-12)), x-move,
                  x+move),
34            xmin, xmax)
35        if xnew.mean() > volfrac: l1 = lmid
36        else:                     l2 = lmid
37    return xnew
```

Listing A.2: SIMP stiffness assembly and compliance solve (*top*); per-element sensitivity and filtered OC update (*bottom*). (Eqs. 3.5, 3.9, 3.16, 3.18).

A.3 MFAS Configuration and Acceptance Gate

```
@dataclass
class MethodCfg:
    method:             str
    f_verify:           int   = 5
    eta_tr:             float = 0.10
    sigma_ratio:        float = 0.0001
    accept_warmup:      int   = 0       # set to 20 for M4
    fine_verify_steps:  int   = 0
    move_fine_verify:   float = 0.10
    min_iter:           int   = 0
    max_iter:           int   = 150
    tol:                float = 1e-3
    fine_cleanup_iters:int   = 15
    fine_cleanup_tol:   float = 1e-3
    snap_iters:         tuple = field(default_factory=tuple)

# --- Acceptance gate (evaluated at each verification event)
   ---
sigma      = float(mc.sigma_ratio)*max(1e-12, abs(C_best_f))
dC_pred    = float(Cc_prev - Cc) if Cc_prev is not None else
   0.0
dC_act     = float(C_f_prev - C_f_new)
near_static= abs(dC_pred) < sigma

if near_static:                                          # Case A
    accept = True
elif it < mc.accept_warmup and mc.method == "M4":   # Case B
    accept = bool(C_f_new <= C_f_prev + sigma)
else:                                                    # Case C
    accept = bool(float(dC_act/dC_pred) >= mc.eta_tr)

if accept:
    x_f_accepted = x_f_prop.copy()
    C_f_prev = C_f_new
    C_best_f = min(C_best_f, C_f_new)
    logs["n_accept"] += 1
else:
    x_c_reset = downsample2x_avg(x_f_accepted)
    x_c       = np.clip(x_c_reset, 0.001, 1.0)
```

```
    xPhys_c    = apply_filter(Hc, Hcs, x_c)
    xPhys_c    = enforce_mean_volume(xPhys_c, vf, xmax=1.0)
    logs["n_reject"] += 1
```

Listing A.3: Run configuration dataclass (*top*) and TR-lite acceptance gate with coarse resynchronisation on rejection (*bottom*). Three cases implement Section 3.4.3 and Algorithm 1.

A.4 Cleanup, U-Net, and M4 Entry

```
def run_fine_cleanup(problem, x_f_init,
                     cleanup_iters=15, cleanup_tol=1e-3):
    KE=element_stiffness_q4(problem.E0, problem.nu)
    edofMat,iK,jK,_,nodenrs=build_edof(problem.nelx_f,problem.nely_f)
    H,Hs=build_filter(problem.nelx_f,problem.nely_f,problem.rmin_f)
    vf=float(problem.volfrac);
       x=np.clip(x_f_init.copy(),0.001,1.0)
    xPhys=apply_filter(H,Hs,x)
    for _ in range(cleanup_iters):
        C,U,_,_=fea_compliance(problem.nelx_f,problem.nely_f,
                               xPhys,problem,KE,edofMat,iK,jK,nodenrs)
        ce=np.sum((U[edofMat]@KE)*U[edofMat],axis=1)
        dc=(-problem.penal*(problem.E0-problem.Emin)
            *xPhys.flatten(order='F')**(problem.penal-1)*ce
           ).reshape((problem.nely_f,problem.nelx_f),order='F')
        x_flat=x.flatten(order='F')
        dcf=((H@(x_flat*dc.flatten(order='F')))
             /(Hs*np.maximum(1e-3,x_flat))
            ).reshape((problem.nely_f,problem.nelx_f),order='F')
        xPhys_old=xPhys.copy(); x=oc_update(x,dcf,vf,move=0.2)
        xPhys=apply_filter(H,Hs,x)
        if
           float(np.max(np.abs(xPhys-xPhys_old)))<cleanup_tol:
           break
    return xPhys

class UNetSmall(nn.Module):
    def __init__(self, in_ch=4, base=32):
        super().__init__()
        def CB(ic,oc):
```

```
            return nn.Sequential(nn.Conv2d(ic,oc,3,padding=1),nn.ReLU(),
                                 nn.Conv2d(oc,oc,3,padding=1),nn.ReLU())
        self.enc1=CB(in_ch,base); self.pool1=nn.MaxPool2d(2)
        self.enc2=CB(base,base*2);self.pool2=nn.MaxPool2d(2)
        self.bott=CB(base*2,base*4)
        self.proj2=nn.Conv2d(base*4,base*2,1)
        self.dec2=CB(base*4,base*2)
        self.proj1=nn.Conv2d(base*2,base,1)
        self.dec1=CB(base*2,base); self.out=nn.Conv2d(base,1,1)
    def forward(self,x):
        e1=self.enc1(x); e2=self.enc2(self.pool1(e1))
        b=self.bott(self.pool2(e2))
        u2=F.interpolate(self.proj2(b),size=e2.shape[2:],
                         mode="bilinear",align_corners=False)
        d2=self.dec2(torch.cat([u2,e2],dim=1))
        u1=F.interpolate(self.proj1(d2),size=e1.shape[2:],
                         mode="bilinear",align_corners=False)
        return torch.sigmoid(self.out(
            self.dec1(torch.cat([u1,e1],dim=1))))

if mc.method == "M4":
    nelx_c, nely_c = problem.nelx_f//2, problem.nely_f//2
    x_c_init = None
    if unet_model is not None:
        try:
            x_c_init = unet_predict_coarse(
                problem, nelx_c, nely_c, unet_model)
            Hc,Hcs = build_filter(nelx_c, nely_c,
                                  max(1.0,problem.rmin_f/2.0))
            x_c_init = np.clip(x_c_init,0.001,1.0)
            x_c_init = apply_filter(Hc,Hcs,x_c_init)
            x_c_init = enforce_mean_volume(x_c_init,problem.volfrac)
        except Exception:
            x_c_init = None
    if x_c_init is None:
        x_c_init = np.full((nely_c,nelx_c), problem.volfrac)
    result, logs = mfas_engine(problem, mc, x_c_init,
```

```
65                                          snap_iters=snap,
                                               verbose=verbose)
```

Listing A.4: Post-MFAS fine-mesh cleanup (*top*); 2D U-Net forward pass (*middle*); M4 warm-start entry point with fallback to uniform density (*bottom*). The decoder uses bilinear `F.interpolate` rather than transposed convolutions; no batch normalisation is applied.

Appendix B

Implementation Snippets: Three-Dimensional Framework

This appendix covers selected implementation excerpts for the three-dimensional MFAS framework, covering the H8 FEM core, PCG solver, and SIMP optimiser (Sections B.1–B.2), and the MFAS-specific components (Sections B.3–B.4).

B.1 H8 Element Stiffness Matrix

```
def element_stiffness_h8(E: float, nu: float) -> torch.Tensor:
    k = 1./math.sqrt(3.); gp = [-k, k]
    C = E/((1+nu)*(1-2*nu)) * np.array([
        [1-nu, nu,   nu,   0,          0,          0
           ],
        [nu,   1-nu, nu,   0,          0,          0
           ],
        [nu,   nu,   1-nu, 0,          0,          0
           ],
        [0,    0,    0,    (1-2*nu)/2, 0,          0
           ],
        [0,    0,    0,    0,          (1-2*nu)/2, 0
           ],
        [0,    0,    0,    0,          0,
           (1-2*nu)/2]])
    Ke = np.zeros((24,24), dtype=np.float64)
    Xn = np.array([[-1,-1,-1],[1,-1,-1],[1,1,-1],[-1,1,-1],
                   [-1,-1, 1],[1,-1, 1],[1,1, 1],[-1,1, 1]])
    for xi in gp:
      for eta in gp:
```

```
        for zeta in gp:
            dN = 0.125*np.array([
              [-(1-eta)*(1-zeta),(1-eta)*(1-zeta),
                (1+eta)*(1-zeta),-(1+eta)*(1-zeta),
               -(1-eta)*(1+zeta),(1-eta)*(1+zeta),
                (1+eta)*(1+zeta),-(1+eta)*(1+zeta)],
              [-(1-xi)*(1-zeta),-(1+xi)*(1-zeta),
                (1+xi)*(1-zeta), (1-xi)*(1-zeta),
               -(1-xi)*(1+zeta),-(1+xi)*(1+zeta),
                (1+xi)*(1+zeta), (1-xi)*(1+zeta)],
              [-(1-xi)*(1-eta),-(1+xi)*(1-eta),
               -(1+xi)*(1+eta),-(1-xi)*(1+eta),
                (1-xi)*(1-eta), (1+xi)*(1-eta),
                (1+xi)*(1+eta), (1-xi)*(1+eta)]])
            J=dN@Xn; dN_dx=np.linalg.inv(J)@dN
            B=np.zeros((6,24))
            for n in range(8):
                B[0,3*n]=dN_dx[0,n]; B[1,3*n+1]=dN_dx[1,n]
                B[2,3*n+2]=dN_dx[2,n]
                B[3,3*n]=dN_dx[1,n]; B[3,3*n+1]=dN_dx[0,n]
                B[4,3*n+1]=dN_dx[2,n]; B[4,3*n+2]=dN_dx[1,n]
                B[5,3*n]=dN_dx[2,n]; B[5,3*n+2]=dN_dx[0,n]
            Ke += (B.T@(C@B))*np.linalg.det(J)
    return torch.tensor(Ke, dtype=torch.float64,
       device=device)
```

Listing B.1: H8 element stiffness matrix via $2 \times 2 \times 2$ Gauss quadrature. The strain-displacement matrix $\mathbf{B}$ is assembled from the physical shape-function derivatives $\mathbf{J}^{-1}\hat{\mathbf{N}}$; $\mathbf{K}_e = \sum_{\text{gp}} \mathbf{B}^\top \mathbf{C}\, \mathbf{B}\, \det \mathbf{J}$.

B.2 PCG Solver, Compliance, Sensitivity, and OC Update

```
def apply_K_free(self, v_free, xPhys):
    v = torch.zeros(self.ndof, dtype=torch.float64,
       device=device)
    v[self.free_mask] = v_free
    ve = v[self.elem_dofs]
    sc = (self.E_min +
       xPhys**self.penal*(1.-self.E_min)).unsqueeze(1)
```

```
6       fe = sc*(ve@self.Ke0.T)
7       Kv = torch.zeros(self.ndof, dtype=torch.float64,
           device=device)
8       Kv.index_add_(0, self.elem_dofs.view(-1), fe.view(-1))
9       return Kv[self.free_mask]
10
11  def solve(self, xPhys, u0=None, tol=1e-6, maxit=400):
12      M_inv =
           1./(self.assemble_diag(xPhys)[self.free_mask]+1e-12)
13      b = self.force[self.free_mask]
14      x = u0[self.free_mask].clone() if u0 is not None \
15          else torch.zeros_like(b)
16      r = b-self.apply_K_free(x,xPhys); z=M_inv*r; p=z.clone()
17      rz=torch.dot(r,z); bnorm=torch.norm(b)+1e-30
18      for it in range(maxit):
19          Ap=self.apply_K_free(p,xPhys);
               alpha=rz/(torch.dot(p,Ap)+1e-30)
20          x+=alpha*p; r-=alpha*Ap
21          if torch.norm(r)/bnorm<=tol: break
22          z=M_inv*r; rz_new=torch.dot(r,z)
23          p=z+(rz_new/(rz+1e-30))*p; rz=rz_new
24      u=torch.zeros(self.ndof, dtype=torch.float64,
           device=device)
25      u[self.free_mask]=x; return u, it+1,
           float(torch.norm(r)/bnorm)
26
27  def solve_equilibrium(self, tol=1e-6, maxit=400):
28      u,nit,res=self.solver.solve(self.xPhys,u0=self.u,tol=tol,maxit=maxit)
29      if nit==maxit or res>1e-3:              # recovery pass
30          u,_,res=self.solver.solve(self.xPhys,u0=u,tol=1e-4,maxit=1200)
31      self.u=u; self.n_fe_solves+=1
32      return float(torch.dot(u,self.force)), nit, res
33
34  def sensitivity(self):
35      ue=self.u[self.elem_dofs];
           ce=(ue*(ue@self.Ke0.T)).sum(dim=1)
36      dE=self.penal*(self.xPhys**(self.penal-1.))*(1.-self.E_min)
37      return -dE*ce                           # dC/d(rho_bar)
38
39  def oc_update(self, dc):
40      xd=(self.xPhys*dc).view(1,1,self.nx,self.ny,self.nz)
```

```
    Hxd=F.conv3d(xd,self.filter_kernel,padding=self.filter_pad).view(-1)
    xp=self.xPhys.view(1,1,self.nx,self.ny,self.nz)
    Hxp=F.conv3d(xp,self.filter_kernel,padding=self.filter_pad).view(-1)
    dc_t=Hxd/(Hxp+1e-12)                    # density-weighted
       filtered sens.
    l1,l2=0.,1e9; x_old=self.x; x_new=x_old.clone()
    for _ in range(60):
        lm=0.5*(l1+l2)
        term=torch.sqrt(torch.clamp(-dc_t/(lm+1e-12),
           min=1e-12))
        cand=torch.clamp(x_old*term, x_old-self.move,
           x_old+self.move)
        cand=torch.clamp(cand, self.rho_min, 1.0)
        if self.passive_mask is not None:
           cand=self.apply_passive(cand)
        vf=self._active_vol(self.apply_density_filter(cand))
        if vf>self.vol_frac: l1=lm
        else: l2=lm; x_new=cand
        if (l2-l1)/(l1+l2+1e-30)<1e-5: break
    self.x=x_new; self.xPhys=self.apply_density_filter(self.x)
    if self.passive_mask is not None:
        self.xPhys=self.apply_passive(self.xPhys)
    return float(self._active_vol(self.xPhys))
```

Listing B.2: Matrix-free PCG matrix-vector product and solver (*top*); compliance evaluation and per-element sensitivity (*middle*); three-dimensional OC update with filtered sensitivity chain (*bottom*). (Eqs. 3.9, 3.17, 3.18).

B.3 U-Net Architecture and MFAS Parameters

```
class UNet3D(nn.Module):
    def __init__(self, in_ch=6, base=16):
        super().__init__()
        self.e1=_CB(in_ch,base); self.p1=nn.MaxPool3d(2)
        self.e2=_CB(base,base*2);self.p2=nn.MaxPool3d(2)
        self.bot=_CB(base*2,base*4)
        self.u2=nn.ConvTranspose3d(base*4,base*2,2,stride=2)
        self.d2=_CB(base*4,base*2)
        self.u1=nn.ConvTranspose3d(base*2,base,2,stride=2)
```

```
        self.d1=_CB(base*2,base);
           self.head=nn.Conv3d(base,1,1)
    def forward(self,x):
        e1=self.e1(x); e2=self.e2(self.p1(e1))
        b=self.bot(self.p2(e2)); d2=self.u2(b)
        if d2.shape[2:]!=e2.shape[2:]:
            d2=F.interpolate(d2,size=e2.shape[2:],
                             mode='trilinear',align_corners=False)
        d2=self.d2(torch.cat([d2,e2],1)); d1=self.u1(d2)
        if d1.shape[2:]!=e1.shape[2:]:
            d1=F.interpolate(d1,size=e1.shape[2:],
                             mode='trilinear',align_corners=False)
        return
           torch.sigmoid(self.head(self.d1(torch.cat([d1,e1],1))))

# MFAS parameters (see Tables 4.3--4.4 for full parameter set)
rmin_f=2.0; rmin_c=1.5              # rmin_c > rmin_f/2 for
   PCG cond.
epsilon_rel=0.010; cleanup_iters=15
verify_freq_M3_cant=5; verify_freq_M3_lb=5
verify_freq_M4_cant=5; verify_freq_M4_lb=5
min_verifications=10; no_improve_limit=5; cbest_conv_tol=1e-3
recovery_iters_lb=25; recovery_move_lb=0.10
unet_lr=1e-3; unet_lr_decay=0.95; unet_batch=2
unet_epochs_cant=150; unet_epochs_lb=80
n_dataset_cant=180; n_dataset_lb=80
```

Listing B.3: 3D U-Net forward pass (*top*) and MFAS-critical parameters (*bottom*). The trilinear fallback handles non-power-of-two coarse-mesh dimensions. U-Net epoch budgets (150 and 80) supersede the nominal figure in Section 3.5.3; see Table 4.7.

B.4 Stage 0: Quality Gate; Stage 1: MFAS Loop; Stage 2: Recovery and Cleanup

```
QUALITY_THRESHOLD=0.95; warm_started=False
if unet_model is not None:
    inp=build_unet_input(prob,coarse)
    with torch.no_grad():
        pred=unet_model(inp.float())
```

```
        if
           pred.shape[2:]!=torch.Size([coarse.nx,coarse.ny,coarse.nz]):
            pred=F.interpolate(pred,
                size=(coarse.nx,coarse.ny,coarse.nz),
                mode='trilinear',align_corners=False)
    rho_c0=pred.squeeze().reshape(-1).double().clamp(rho_min,1.)
    if coarse.passive_mask is not None:
        rho_c0=coarse.apply_passive(rho_c0)
    coarse.xPhys=rho_c0; coarse.x=rho_c0.clone()
    enforce_volume(coarse,coarse.vol_frac)
    with torch.no_grad():
        xf=prolong_c2f(coarse.xPhys,
                       (coarse.nx,coarse.ny,coarse.nz),
                       (fine.nx,fine.ny,fine.nz))
        fine.xPhys=xf; fine.x=xf.clone()
        if fine.passive_mask is not None:
            fine.xPhys=fine.apply_passive(fine.xPhys)
        enforce_volume(fine,fine.vol_frac)
    C_unet,_,_=fine.solve_equilibrium(pcg_tol,pcg_maxit)
    fine_cold=make_topopt(prob,'fine',seed)
    C_uniform,_,_=fine_cold.solve_equilibrium(pcg_tol,pcg_maxit)
    if C_unet<QUALITY_THRESHOLD*C_uniform:
        warm_started=True; C_best=float(C_unet)
        xP_best=fine.xPhys.clone(); u_best=fine.u.clone()
    else:
        coarse_cold=make_topopt(prob,'coarse',seed)
        fine.xPhys=fine_cold.xPhys.clone()
        fine.x=fine_cold.x.clone(); fine.u=fine_cold.u.clone()
        coarse.xPhys=coarse_cold.xPhys.clone()
        coarse.x=coarse_cold.x.clone()
        C_best=float(C_uniform)
        xP_best=fine.xPhys.clone(); u_best=fine.u.clone()
```

Listing B.4: Stage 0 U-Net quality gate (*top*). One fine-mesh FEA on the prolongated U-Net prediction yields C_{unet}; acceptance requires $C_{\text{unet}} < 0.95\, C_{\text{uniform}}$. Failure resets all state to the cold uniform start, making M4 equivalent to M3 for that run. Without this guard, the coarse resynchronisation at `it` $= 0$ would overwrite `coarse.xPhys` with the restriction of the uniform fine-mesh state, discarding the U-Net warm-start density before the first coarse iteration.

```
for it in range(max_iter):
    coarse.move=move_schedule(it)
```

```
    if it%f_verify==0:
        if it==0 and warm_started:
            coarse.u=torch.zeros_like(coarse.u) # preserve U-Net init
        else:
            with torch.no_grad():
                coarse.xPhys=restrict_f2c(xP_best,
                    (fine.nx,fine.ny,fine.nz),
                    (coarse.nx,coarse.ny,coarse.nz))
                coarse.x=coarse.xPhys.clone()
                if coarse.passive_mask is not None:
                    coarse.xPhys=coarse.apply_passive(coarse.xPhys)
            coarse.u=torch.zeros_like(coarse.u)
    Cc,_,_=coarse.solve_equilibrium(pcg_tol,pcg_maxit)
    dc=coarse.sensitivity(); coarse.oc_update(dc)
    if (it+1)%f_verify==0:
        with torch.no_grad():
            fine.xPhys=prolong_c2f(coarse.xPhys,
                (coarse.nx,coarse.ny,coarse.nz),
                (fine.nx,fine.ny,fine.nz))
            fine.x=fine.xPhys.clone()
            if fine.passive_mask is not None:
                fine.xPhys=fine.apply_passive(fine.xPhys)
            enforce_volume(fine,fine.vol_frac)
        Cf,_,_=fine.solve_equilibrium(pcg_tol,pcg_maxit)
        if Cf<=(1.+eps_rel)*C_best:
            npass+=1
            if first_accept_iter is None: first_accept_iter=it
            if Cf<C_best:
                C_best=Cf; xP_best=fine.xPhys.clone()
                u_best=fine.u.clone(); no_imp=0
            else: no_imp+=1
        else:
            nfail+=1; no_imp+=1
            fine.xPhys=xP_best; fine.x=xP_best.clone()
            fine.u=u_best
        nv=npass+nfail
        if no_imp>=no_improve_limit and
           nv>=min_verifications: break
        if nv>=min_verifications and
           len(C_best_hist)>no_improve_limit:
```

```
            c_ref=C_best_hist[-(no_improve_limit+1)]
            if (c_ref-C_best)/(c_ref+1e-30)<cbest_conv_tol:
                break
        C_best_hist.append(C_best)

if prob=='lb' and recovery_iters_lb>0:
    fine.xPhys=xP_best; fine.x=xP_best.clone(); fine.u=u_best
    for _ in range(recovery_iters_lb):
        fine.move=recovery_move_lb
        C,_,_=fine.solve_equilibrium(pcg_tol,pcg_maxit)
        fine.oc_update(fine.sensitivity())
    xP_best=fine.xPhys.clone(); u_best=fine.u.clone()

fine.xPhys=xP_best; fine.x=xP_best.clone(); fine.u=u_best
xp_prev=fine.xPhys.clone(); consec=0
for _ in range(cleanup_iters):
    fine.move=move_ultrafine
    C,_,_=fine.solve_equilibrium(pcg_tol,pcg_maxit)
    fine.oc_update(fine.sensitivity())
    ch=fine.get_density_change(xp_prev);
       xp_prev=fine.xPhys.clone()
    consec=consec+1 if ch<tol_change else 0
    if consec>=3: break
C_final=C
```

Listing B.5: Stage 1 MFAS loop with iteration-zero guard (*top*) and Stage 2 L-bracket recovery plus fine-mesh cleanup (*bottom*). The guard preserves the U-Net coarse density at iteration zero; in M3 it is never activated. Acceptance: $C_{\text{trial}} \leq (1+\epsilon_{\text{rel}})C_{\text{best}}$ (Eq. 3.33, Algorithm 2). Recovery (25 iters, move = 0.10) escapes the coarse-mesh local minimum at the L-bracket re-entrant corner. Cleanup exits after three consecutive iters below `tol_change` = 0.01.

Appendix C

Supplementary Quantitative Results

This appendix provides two categories of data not present in Chapter 5 tables: stage-wise wall-clock decompositions for 2D and 3D, and individual per-seed records for the 3D experiments.

C.1 Stage-Time Decompositions

Table C.1: Mean stage-time decomposition for the 2D experiments, averaged over $n = 5$ seeds per (problem, method) cell. $\bar{t}_{\text{fine}}$ = fine-mesh FEA time (s); $\bar{t}_{\text{coarse}}$ = coarse-mesh FEA time (s); $\bar{t}_{\text{cl}}$ = Stage 2 fine-mesh cleanup time (s); $\bar{t}_{\text{total}}$ = total wall-clock time (s). For M1 and M2, cleanup is not applied; M2 coarse and fine FEA times are equal because a single coarse SIMP run feeds one fine prolongation.

Problem	Method	$\bar{t}_{\text{fine}}$	$\bar{t}_{\text{coarse}}$	$\bar{t}_{\text{cl}}$	$\bar{t}_{\text{total}}$
MBB	M1	54.07	0.00	0.00	55.69
	M2	7.48	7.48	0.00	8.59
	M3	9.89	8.91	6.15	26.21
	M4	10.03	8.91	6.11	26.36
Tip Cantilever	M1	69.31	0.00	0.00	71.36
	M2	9.68	9.68	0.00	10.93
	M3	13.03	11.50	6.09	32.04
	M4	13.27	11.44	6.01	32.17
Mid Cantilever	M1	69.46	0.00	0.00	71.50
	M2	9.32	9.32	0.00	10.56
	M3	12.44	11.08	6.12	31.09
	M4	12.68	11.30	6.17	31.48

Table C.2: Mean stage-time decomposition for the 3D experiments, averaged over $n = 3$ seeds. $\bar{t}_{\text{unet}}$ = U-Net inference and quality-gate evaluation (M4 only; includes the two-FEA quality check); $\bar{t}_{\text{MFAS}}$ = Stage 1 coarse iterations and fine verification calls; $\bar{t}_{\text{rec}}$ = post-MFAS fine-mesh recovery (L-bracket only); $\bar{t}_{\text{cl}}$ = Stage 2 fine-mesh cleanup.

Problem	Method	$\bar{t}_{\text{unet}}$	$\bar{t}_{\text{MFAS}}$	$\bar{t}_{\text{rec}}$	$\bar{t}_{\text{cl}}$	$\bar{t}_{\text{total}}$
Cantilever	M1	—	—	—	—	117.9
	M2	—	—	—	—	40.2
	M3	0.0	64.6	0.0	15.1	79.7
	M4	1.8	75.9	0.0	15.1	92.8
L-bracket	M1	—	—	—	—	78.0
	M2	—	—	—	—	48.7
	M3	0.0	89.5	17.6	10.6	117.7
	M4	1.2	33.2	17.6	10.6	62.6

C.2 Three-Dimensional Per-Seed Records

The tables below record each seed individually. For the cantilever, seeds encode load-patch z-offsets $\delta_z \in \{-1, 0, +1\}$ (seeds 1, 2, 3 respectively), introducing genuine compliance variability via PCG convergence interaction with the load geometry. For the L-bracket, load position is seed-independent and variability is negligible.

Table C.3: 3D cantilever ($180 \times 60 \times 44$ fine, $v_f = 0.10$) — per-seed results. C = post-cleanup fine-mesh compliance. t = total wall-clock time (s). $\hat{\kappa}$ = MFAS acceptance rate. N_f/N_c = fine-/coarse-mesh FEA call counts. γ = final grey fraction.

Method	Seed	C	t (s)	$\hat{\kappa}$	N_f	N_c	γ
M1	1	12.106	122.7	—	—	—	0.1033
M1	2	12.050	107.4	—	—	—	0.1035
M1	3	12.106	123.4	—	—	—	0.1033
M2	1	14.540	44.7	—	—	—	0.1576
M2	2	16.603	37.9	—	—	—	0.1817
M2	3	16.743	38.1	—	—	—	0.1817
M3	1	12.014	49.1	0.500	26	50	0.0927
M3	2	12.468	108.4	1.000	45	145	0.0991
M3	3	12.512	81.8	1.000	36	100	0.1001
M4	1	12.528	101.5	1.000	41	125	0.0987
M4	2	12.479	86.1	1.000	37	105	0.0980
M4	3	12.521	90.8	1.000	38	110	0.0989

Table C.4: 3D L-bracket ($90 \times 90 \times 30$ fine, $v_f = 0.15$) — per-seed results. Column definitions as in Table C.3. The 25-iteration recovery phase accounts for the difference between M3/M4 total time and the Stage 1 + cleanup time visible in Table C.2.

Method	Seed	C	t (s)	$\hat{\kappa}$	N_f	N_c	γ
M1	1	22.793	77.4	—	—	—	0.1071
M1	2	22.782	79.1	—	—	—	0.1072
M1	3	22.794	77.5	—	—	—	0.1071
M2	1	37.591	48.8	—	—	—	0.1836
M2	2	37.591	48.9	—	—	—	0.1836
M2	3	37.591	48.6	—	—	—	0.1836
M3	1	22.516	117.9	1.000	71	150	0.0875
M3	2	22.516	117.9	1.000	71	150	0.0875
M3	3	22.516	117.4	1.000	71	150	0.0875
M4	1	22.667	62.6	1.000	52	55	0.0922
M4	2	22.667	62.6	1.000	52	55	0.0922
M4	3	22.667	62.5	1.000	52	55	0.0922

Appendix D

Complete Statistical Test Results

This appendix collects the full 2D paired test table (per-problem breakdown with all test statistics) and a minimum-detectable-effect analysis that quantifies what the observed $n = 3$ seed count implies for future experiment design.

D.1 Two-Dimensional Runtime Tests

Since $\sigma_C = 0$ in 2D (Section 5.1.3), all paired tests apply to wall-clock time. Table D.1 reports per-seed speedup distributions; Table D.2 reports the Wilcoxon signed-rank test results. With $n = 5$ and all differences positive, $W = 15$ (maximum possible) and $p = 0.0156$ (two-tailed) for every comparison shown.

Table D.1: Two-dimensional per-seed speedup distribution ($S_i = t_{\mathrm{M1},i}/t_{\mathrm{method},i}$, $n = 5$ seeds). $\bar{S} \pm \hat{s}_S$ = mean ± SD; S_{50} = median; $S_{\min}$, $S_{\max}$ = range.

Problem	Method	$\bar{S} \pm \hat{s}_S$	S_{50}	$S_{\min}$	$S_{\max}$
MBB	M2	6.492 ± 0.091	6.470	6.388	6.603
	M3	2.127 ± 0.033	2.123	2.092	2.178
	M4	2.116 ± 0.031	2.109	2.086	2.170
Tip Cantilever	M2	6.537 ± 0.196	6.525	6.251	6.778
	M3	2.229 ± 0.054	2.232	2.149	2.299
	M4	2.219 ± 0.044	2.210	2.168	2.280
Mid Cantilever	M2	6.774 ± 0.131	6.747	6.677	6.947
	M3	2.302 ± 0.046	2.303	2.262	2.342
	M4	2.280 ± 0.057	2.274	2.207	2.379

Table D.2: Two-dimensional Wilcoxon signed-rank test results for wall-clock speedup (two-tailed, H_1: method faster than M1; $n = 5$ paired seeds per cell). $W = 15$ is the maximum attainable value (all five pairs in the hypothesised direction); $p = 0.0156$ is exact. $d_z = (\bar{S} - 1)/\hat{s}_S$ quantifies the standardised speedup effect. M1 is the reference; M2 is included for completeness.

Problem	Method vs. M1	W	p	d_z
MBB	M2	15	0.0156	60.02
	M3	15	0.0156	34.16
	M4	15	0.0156	36.65
Tip Cantilever	M2	15	0.0156	28.24
	M3	15	0.0156	21.47
	M4	15	0.0156	27.71
Mid Cantilever	M2	15	0.0156	44.46
	M3	15	0.0156	28.30
	M4	15	0.0156	22.46

All Cohen's d_z values are very large, reflecting tight within-method timing variance ($\hat{s}_S < 0.20$) against speedup means well above unity.

D.2 Minimum Detectable Effect and Power

As noted in Section 5.6, the 3D ensemble of $n = 3$ seeds imposes a hard statistical power limit: for the one-tailed Wilcoxon signed-rank test at $\alpha = 0.05$, the minimum achievable p-value is 0.125, so no comparison can reach significance regardless of the true effect size.

Table D.3 translates this limitation into minimum detectable effect sizes (MDE) for a paired t-test at $\alpha = 0.05$ (one-tailed) and power $1 - \beta = 0.80$. At $n = 3$, the MDE is $d_z \approx 2.73$—only effects larger than the M2-versus-M1 compliance gap on the L-bracket would be formally detectable. Extending to $n \geq 10$ lowers the MDE to the large-effect range ($d_z \approx 1.06$), which is sufficient to detect the compliance differences observed between M3/M4 and M1 on the cantilever.

Table D.3: Minimum detectable effect (d_z, Cohen's standardised mean paired difference) for a one-tailed paired t-test at $\alpha = 0.05$ and power $1 - \beta = 0.80$. Current 3D design: $n = 3$.

n	MDE (d_z)	Effect size label
3	2.73	Very large
5	1.77	Very large
8	1.23	Large
10	1.06	Large
15	0.83	Large
20	0.70	Medium–Large
30	0.56	Medium

The practical implication is that compliance and runtime differences reported in Table 5.2 represent real observed outcomes in the tested sample. Confidence intervals on $\bar{C}$ (also in Table 5.2) provide a more informative summary than binary significance outcomes at this sample size, and should be the primary basis for interpreting the 3D results.